UNIVERSITÉ
DE GENÈVE

Master's Thesis

Système de visualisation, d'annotation et de transcription des manuscrits numérisés de Ferdinand de Saussure

---

Massimo BRERO

Septembre 2013

Encadrant :
Stéphane MARCHAND-MAILLET

Co-encadrants :
Gilles FALQUET
Luka NERIMA

Table des matières

Table des matières **2**

Remerciements **7**

Résumé **8**

1 Introduction **9**
1.1 Problématique générale . . . 9
1.2 Problématique du travail . . . 10

2 Analyse des besoins **11**
2.1 Déroulement et méthodologie . . . 11
2.2 Synthèse des spécifications . . . 12
2.2.1 Visualisation . . . 12
2.2.2 Édition d'annotations/transcriptions/métadonnées . . . 12
2.2.3 Stockage . . . 13
2.2.4 Mise à jour du contenu du site . . . 13
2.2.5 Recherche d'information . . . 14
2.2.6 Navigation . . . 14

3 État de l'art **15**
3.1 Projets *digital humanities* similaires . . . 15
3.1.1 Quelques projets représentatifs du domaine . . . 16
3.1.2 Nietzsche Source . . . 17
3.1.3 Utrecht University Library Special Collections . . . 18
3.1.4 Wellcome Digital Library . . . 18
3.1.5 Bentham Project . . . 18
3.1.6 Bilan intermédiaire . . . 19
3.2 Format des annotations/transcriptions et catalogage . . . 20
3.2.1 METS . . . 20
3.2.2 TEI Recommandations . . . 20
3.2.2.1 Manuscript Description . . . 21
3.2.2.2 Representation of Primary Sources . . . 21
3.2.3 Bilan intermédiaire . . . 21
3.3 Triplestore / Quadstore . . . 22
3.3.1 Stockage des métadonnées et des annotations . . . 22
3.3.2 RDF/SPARQL . . . 23
3.3.3 Triplestore et *endpoint* SPARQL . . . 24
3.3.3.1 Choix du triplestore . . . 24
3.3.3.2 Moteur de recherche / Indexation / Full text search . . . 24
3.4 Affichage d'images en haute résolution . . . 25
3.4.1 GIS (*Geographic information system*) . . . 26
3.4.1.1 Google Maps (with Google Maps API) . . . 26
3.4.1.2 Esri ArcGIS . . . 26
3.4.1.3 Old Maps Online . . . 27
3.4.2 Serveur d'images / Création de *tiles* . . . 27
3.4.2.1 Zoomify . . . 27
3.4.2.2 Deep Zoom . . . 28

3.4.2.3 djatoka (aDORe djatoka) . . . 28
3.4.2.4 IIPImage . . . 28
3.4.3 Viewers (et possibilités d'annotations) . . . 30
3.4.3.1 Viewers non retenus . . . 30
3.4.3.2 Zoomify (Flash + Zoomify) . . . 31
3.4.3.3 IIPZoom (Flash) . . . 32
3.4.3.4 IIPMooViewer (Javascript + IIP/djatoka/Zoomify/Deep Zoom) 32
3.4.3.5 OpenLayers (Javascript + Zoomify) . . . 33
3.4.4 Projets informatiques similaires . . . 33
3.4.4.1 BAMBI . . . 34
3.4.4.2 T-Pen . . . 34
3.4.4.3 Diva.js . . . 34
3.4.4.4 Lincolnnus/Image-Viewer . . . 35
3.4.5 Bilan intermédiaire . . . 35
3.4.5.1 Serveurs . . . 35
3.4.5.2 Viewers et projets similaires . . . 37
3.5 Images . . . 38
3.5.1 Problématique . . . 38
3.5.2 *Tiles* (Tuiles) . . . 38
3.5.3 Compression *lossless* or *lossy* . . . 42
3.5.3.1 Quand compresser et quelle compression utiliser ? . . . 43
3.5.4 Formats d'image . . . 43
3.5.4.1 TIFF . . . 43
3.5.4.2 PTIF (Pyramid TIFF) . . . 44
3.5.4.3 JPEG2000 . . . 44
3.5.4.4 *Tiles* statiques (arborescence) . . . 48
3.5.4.5 Test comparatif . . . 49
3.5.5 Bilan : comparaison TIFF - JPEG2000 - *tiles* statiques . . . 50

4 Modèle de données **52**
4.1 Concepts du domaine . . . 52
4.1.1 Annotations . . . 52
4.1.2 Transcriptions . . . 53
4.1.3 Identification des images de manuscrits (cote) . . . 55
4.2 UML et explication du modèle . . . 57
4.3 RDFS . . . 58
4.4 RDF et génération d'URI uniques . . . 60

5 Interfaces utilisateurs **63**
5.1 Premiers prototypes . . . 63
5.1.1 Premier prototype . . . 63
5.1.2 Second prototype . . . 65
5.1.3 Troisième prototype . . . 66
5.2 Prototype final . . . 67
5.2.1 Application client . . . 67
5.2.2 Accéder aux manuscrits : recherche et navigation . . . 74
5.2.2.1 Recherche . . . 74
5.2.2.2 Navigation . . . 75

6 Implémentation **78**
6.1 Architecture et déploiement . . . 78
6.1.1 Architecture de l'application globale . . . 79
6.1.2 *Workflow* du point de vue du client . . . 80
6.1.3 Déploiement des systèmes . . . 81
6.2 Authentification et sécurisation du serveur . . . 81
6.2.1 Authentification avec UserCake . . . 82
6.2.2 Sécurisation du triplestore OpenRDF Sesame (Tomcat) . . . 82
6.3 Auto alimentation du système (Python) . . . 83
6.3.1 Corpus d'images : problème de conversion et images de mauvaise qualité . 83
6.3.2 Fonctionnement du script Python . . . 84
6.3.3 Convention de nommage . . . 86
6.3.4 Quelques cas d'extraction de cotes détaillés . . . 86
6.3.4.1 Exemples de cotes acceptées . . . 86
6.3.4.2 Exemples de cotes ambiguës . . . 87
6.3.4.3 Exemples de cotes refusées . . . 88
6.4 Backend (PHP) . . . 88
6.4.1 Accès au *endpoint* SPARQL (librairie/cURL) . . . 89
6.4.2 Construction et exécution des requêtes SPARQL . . . 90
6.4.3 Affichage et mise en forme des réponses des requêtes . . . 90
6.5 Recherche et navigation dans les archives . . . 90
6.5.1 Navigation dans les archives . . . 90
6.5.1.1 Algorithme pour la navigation . . . 91
6.5.2 Moteur de recherche . . . 92
6.5.2.1 Algorithme de recherche . . . 92
6.5.2.2 Fonctionnement du moteur de recherche . . . 94
6.5.2.3 Problème de *word boundaries* en Unicode . . . 96
6.5.2.4 Regex et HTML . . . 97
6.5.2.5 Coloration des termes de recherche . . . 98
6.6 Application client/serveur - Service frontal (PHP) . . . 102
6.6.1 Algorithme du service web (frontal) . . . 102
6.7 Application client/serveur - Client (HTML/Javascript) . . . 105
6.7.1 Contraintes de développement en Javascript . . . 105
6.7.2 Liste des fichiers . . . 106
6.7.3 Extensions IIPMooViewer . . . 106
6.7.3.1 Lincolnnus/Image-Viewer . . . 107
6.7.3.2 extended-toolbar.js . . . 107
6.7.3.3 annotations-edit-mass.js . . . 109
6.7.3.4 navigation_slider.js . . . 113
6.7.4 Application client personnalisée . . . 114
6.7.4.1 iipmooviewer-2.0.js . . . 114
6.7.4.2 concepts_saussuriens.js . . . 115
6.7.4.3 iip_semantic_web_brero.js . . . 116
6.7.4.4 mass_editor.js . . . 117
6.7.4.5 addTinyMCE.js . . . 119
6.8 Exemple d'une action détaillée . . . 119
6.8.1 Ajout d'une annotation . . . 119

7 Test d'utilisabilité **126**
7.1 Méthodologie . . . 126
7.1.1 Nombre de participants . . . 126
7.1.1.1 Recherche d'erreurs ou de problèmes : 5 utilisateurs . . . 126
7.1.1.2 Preuve d'utilisabilité : autant d'utilisateurs que possible . . . 127
7.1.1.3 Les participants . . . 127
7.1.2 Méthode d'évaluation . . . 128
7.1.2.1 Conditions d'évaluation . . . 128
7.1.2.2 Protocole de test . . . 129
7.2 Résultats des évaluations . . . 130
7.2.1 Efficacité . . . 130
7.2.2 Efficience . . . 132
7.2.3 Satisfaction (*System Usability Scale* (SUS)) . . . 136
7.2.4 Commentaires, propositions d'amélioration, remarques des participants . 139
7.3 Bilan . . . 140
7.3.1 Modifications/Améliorations proposées . . . 141

8 Conclusion **142**
8.1 Travaux futurs . . . 142
8.2 Conclusion . . . 144

A Cahier des charges **146**
A.1 Version 1 . . . 146
A.2 Version 2 . . . 147

B Script et code **152**
B.1 Conversion d'image . . . 152
B.1.1 Conversion d'un fichier TIFF en PTIF . . . 152
B.1.2 Conversion d'un fichier TIFF en JPEG2000 . . . 152
B.2 Procédures d'installation . . . 153
B.2.1 Installation d'IIPMooViewer 2.0beta . . . 153
B.2.2 Procédure d'installation d'un système Debian . . . 153
B.2.3 Installation de IIP Server, Tomcat, OpenRDF Sésame . . . 153
B.3 Documentation pour l'installation des Triplestores . . . 154
B.3.1 Allegrograph . . . 154
B.3.2 Sesame . . . 154
B.3.3 Virtuoso . . . 155
B.4 Requêtes SPARQL . . . 157
B.4.1 SELECT . . . 157
B.4.2 INSERT . . . 158
B.4.3 DELETE . . . 158
B.4.4 DELETE/INSERT/WHERE . . . 158
B.5 Extraits complets . . . 159
B.5.1 Extrait du fichier de log généré par *auto_alimentation_4.py* . . . 159
B.6 Formulaires HTML supplémentaires - non intégrés . . . 160
B.6.1 Script de correction des noms de fichiers erronés . . . 160
B.6.2 Organisateur de transcriptions . . . 163

C Notes, résumés et recherches annexes **165**
C.1 RDF/Triplestore related . . . 165
C.1.1 Liste de triplestores, d'outils et de librairies . . . 165
C.1.2 Recherche d'informations dans le triplestore . . . 166
C.2 Modification synchrone du triplestore / Accès concurrent et atomicité en PHP . 167

D Documentation technique **169**
D.1 Accès au *endpoint* SPARQL (librairie/cURL) . . . 169
D.1.1 Description des fonctions de sparql_php_curl_lib.php . . . 169
D.2 Construction et exécution des requêtes SPARQL . . . 169
D.2.1 Liste des fonctions de sparql_request_pool.php . . . 169
D.2.2 Description des fonctions de sparql_request_pool.php . . . 170
D.3 Affichage et mise en forme des réponses des requêtes . . . 172
D.3.1 Liste des fonctions de display_sparql_lib.php . . . 172
D.3.2 Description des fonctions de display_sparql_lib.php . . . 172

E Evaluation de l'utilisabilité **175**
E.1 Matériel d'évaluation . . . 175
E.2 Protocole de test . . . 175
E.2.1 Déroulement du test . . . 176
E.2.2 Consentement de participation . . . 177
E.2.3 Formulaire sociodémographique et formulaire de compétences . . . 178
E.2.4 Formulaire SUS . . . 179
E.2.5 Scénario 1 . . . 180
E.2.6 Scénario 2 . . . 181
E.2.7 Scénario 3 . . . 182
E.2.8 Scénario 4 . . . 183

Remerciements

Je tiens à remercier les personnes suivantes de leur soutien ou pour leur participation (de manière directe ou indirecte) dans l'accomplissement de ce travail :
Gaël Boquet, Pierre-Yves Testenoire, Marie Ruchonet, Thalia Brero, Arnaud Béguin, Genoveva Puskas Nerima, Stéphanie Ginalski, Claire Forel, Frédérique Berthelot, Anamaria Bentea, Emmanuel Ducry, Barbara Roth, Gabriela Soare, Christopher Laenzlinger.

Je tiens également à remercier Gilles Falquet, Stéphane Marchand-Maillet et Luka Nerima, les professeurs qui m'ont suivi durant ce travail, pour leur encadrement et leur confiance. Merci tout particulièrement à Gilles pour sa disponibilité et ses suggestions toujours éclairées.

Ce travail n'aurait jamais vu le jour sans les linguistes Daniele Gambarara et Claire Forel, les "saussuriens fous", que je remercie pour leurs conseils, les nombreuses réunions, leur passion et leur inépuisable enthousiasme.

Merci à Arnaud, Gaël, Jérémy, Paul, Pierre et Xavier, mes amis et collègues d'HEPIA.

Un grand merci aussi à Cédric Pénas et Jérémy Gobet pour leurs conseils avisés, ainsi qu'à Corentin pour ses contributions.

Ces remerciements seraient incomplets si je n'en adressais pas à l'ensemble des personnes qui ont participé aux évaluations, pour le temps qu'elles m'ont consacré et les données précieuses qu'elles m'ont permis de récolter.

Merci encore infiniment à Alexia pour son entrain, sa générosité et son soutien tant moral que logistique.

Finalement, je tiens à remercier ma famille pour sa patience et son soutien affectif, et plus particulièrement Thalia et Carolyn qui ont eu la tâche ingrate de relire ce document !

Résumé

Le sujet de ce mémoire de Master est l'étude des différentes options pour concevoir, créer et finalement déployer un système de visualisation, d'annotation et de transcription des manuscrits numérisés du linguiste Ferdinand de Saussure. En effet, les archives des manuscrits de ce dernier sont difficilement accessibles ; c'est pourquoi leur visibilité en ligne s'avère de la plus haute importance pour les linguistes saussuriens. La présentation d'une version numérique de ces archives, sous la forme de photographies en haute résolution, a donc pour vocation première de permettre aux chercheurs de travailler directement sur l'image du manuscrit, mais elle permet également au public intéressé d'avoir accès à cette documentation. Aucun outil équivalent n'existe, ou ils ne sont ni satisfaisants, ni adaptés aux besoins des utilisateurs de ce projet.

Une collaboration étroite avec des linguistes a permis de proposer un modèle de données ouvert offrant la possibilité de publier les données sur le web sémantique (RDF), puis de développer une application complète permettant d'afficher en ligne, et de manière fluide, des images en haute résolution des manuscrits. Une interface graphique aboutie permet d'annoter et de transcrire les manuscrits en travaillant directement sur l'image. La recherche d'information est possible grâce à un moteur de recherche développé selon les besoins des saussuriens.

L'application réalisée a été évaluée par un nombre important d'utilisateurs. Les résultats de ces évaluations, pour le moins positifs, ont permis de prouver que l'application répond aux trois critères d'utilisabilité (efficacité, efficience et satisfaction). Ceci implique que les fonctionnalités souhaitées par les utilisateurs ont été réalisées, mais également que leur apprentissage et leur exécution sont réalisables en un laps de temps raisonnable.

Chapitre 1

Introduction

1.1 Problématique générale

La numérisation et la mise à disposition via internet de manuscrits, peintures, cartes anciennes, etc. sont devenues monnaie courante. En effet, la fragilité, l'unicité et/ou la rareté de ces pièces rend leur accès difficile et requiert souvent des autorisations spéciales afin de pouvoir les consulter et les étudier. Dès lors, un nombre croissant de bibliothèques, d'archives ou de musées recourent à la numérisation de ces objets sous la forme de photographies en haute résolution. L'accès à l'objet original n'est pas nécessaire pour la plupart des chercheurs ou des curieux, car la visualisation d'une version numérisée est suffisante. L'objet physique est donc mieux conservé, et il n'est plus nécessaire de se déplacer pour le consulter, car il est directement visible sur internet. L'étude de l'objet numérisé est même parfois plus aisée de cette façon, car il est possible de zoomer et d'appliquer des traitements sur l'image (contraste, luminosité...) qui permettent de voir des détails qui seraient invisibles à l'œil nu.

La Bibliothèque de Genève (BGE) possède une grande quantité de manuscrits de Ferdinand de Saussure[1]. Ceux-ci représentent une mine d'or non seulement pour la communauté saussurienne, mais également pour des linguistes du monde entier. En effet, Ferdinand de Saussure n'a quasi rien publié et l'accès aux nombreux manuscrits qu'il a écrit offrent aux spécialistes un contenu nouveau à étudier, mais aussi la possibilité de vérifier la validité de transcriptions existantes et controversées.

Ces manuscrits sont en cours de numérisation sous forme de photographies en haute résolution, afin de préserver les originaux et de pouvoir les rendre accessibles au public, mais surtout à la communauté saussurienne. En effet, dans le courant des années quatre-vingts, une consultation trop fréquente a fortement dégradé l'état de ces documents, et leur accès est depuis interdit au public. L'usure de certains feuillets est telle que le papier est devenu presque friable. Ainsi, les photographies numériques sont désormais, pour la plupart des chercheurs, le seul moyen d'étudier le manuscrit.

1. Ferdinand de Saussure (1857-1913) est considéré comme l'un des pères de la linguistique moderne `http://fr.wikipedia.org/wiki/Ferdinand_de_Saussure`. Il ne doit pas être confondu avec son parent le naturaliste et alpiniste Horace-Bénédict `http://fr.wikipedia.org/wiki/Horace-Bénédict_de_Saussure`

1.2 Problématique du travail

Le but de ce travail de Master est de proposer, puis de réaliser un outil scientifique original pour linguistes saussurien, en se basant sur une analyse des besoins de ces derniers. En effet, les spécialistes de Ferdinand de Saussure souhaitent un système leur permettant à la fois de consulter les manuscrits et de pouvoir travailler sur la version numérique des manuscrits.

A moyen/long terme, il est prévu que tous les manuscrits numérisés de Ferdinand de Saussure soient accessibles en ligne sur un site web. Celui-ci permettra au grand public de les consulter. Une interface spéciale donnera la possibilité aux chercheurs de travailler sur le manuscrit pour l'annoter ou le transcrire, ou encore de visualiser plusieurs manuscrits simultanément pour les comparer, les analyser conjointement, etc.

Cette thèse de Master se propose de réaliser une première partie de ce travail. Elle se focalise sur l'analyse des besoins des utilisateurs, la définition des spécifications du projet et la conception d'un premier prototype (application web). Ce dernier se basera sur un état de l'art exhaustif, passant en revue les projets existants dans le domaine des *digital humanities* [2] et les technologies nécessaires à l'aboutissement du projet. Il permettra l'affichage des manuscrits, ainsi qu'une fonctionnalité pour les annoter graphiquement via une interface utilisateur. Celle-ci sera évaluée grâce à des tests réalisés avec des utilisateurs selon un protocole rigoureux. Une chaîne de traitement (*workflow*) permettra de facilement alimenter le contenu du site en automatisant l'ajout de nouvelles images de manuscrits.

Un modèle de données original permettra de structurer l'information. Les transcriptions, les annotations, ainsi que toutes les métadonnées seront stockées sous la forme de triplets RDF [3], dans un triplestore [4], permettant ainsi leur publication sur le web sémantique. Le contenu des annotations et des transcriptions permettra d'étendre les fonctionnalités du site avec des possibilités d'indexation et de recherche d'information.

La visualisation en ligne des manuscrits pose une problématique importante, car les images sont des fichiers volumineux auxquels on doit pouvoir accéder aisément via un navigateur web.

Comme le prototype réalisé dans le cadre de ce Master sera le point de départ d'un système plus ambitieux, une analyse aboutie des besoins des utilisateurs et une bonne compréhension du projet global s'avèrent nécessaires avant la conception du prototype.

Comme aucun système équivalent n'existe, ce projet propose le développement d'un outil scientifique complet et unique dans le domaine des *digital humanities*. Son originalité réside dans le concept de l'outil en tant que tel et ses fonctionnalités (annotation et transcription directement sur l'image grâce à une interface utilisateur intuitive), dans le modèle de données proposé, mais également dans la publication des données du système sur le web sémantique.

2. `http://en.wikipedia.org/wiki/Digital_humanities`
3. `http://en.wikipedia.org/wiki/Resource_Description_Framework`
4. `http://en.wikipedia.org/wiki/Triplestore`

Chapitre 2

Analyse des besoins

2.1 Déroulement et méthodologie

L'ingénierie logicielle, et plus particulièrement l'analyse des besoins des utilisateurs, représente une partie importante de ce travail. En effet, toutes les étapes de la gestion de projet ont dû être prises en charge, de l'analyse des besoins des utilisateurs, jusqu'à la validation, en passant par la définition du cahier des charges, la conception, la réalisation et les tests.

Dans un premier temps, plusieurs entretiens avec les utilisateurs (les saussuriens) ont permis de comprendre et de définir leurs besoins et leurs attentes. Rapidement, les premiers prototypes d'interfaces utilisateurs ont été proposés (voir le chapitre 5). À ce stade du projet, les prototypes pouvaient être de simples pages web statiques, des interfaces *mock*, voire des dessins faits à la main. Au fur et à mesure des entretiens suivants, les besoins et les prototypes ont été discutés, éventuellement adaptés, ou étoffés selon les commentaires et les souhaits des utilisateurs (méthode en spirale).

L'analyse des besoins a été un travail de longue haleine qui s'est presque étendu jusqu'au terme du projet. Comme l'un des utilisateurs réside et travaille à l'étranger, les entretiens étaient souvent espacés dans le temps. De plus, une omission involontaire des utilisateurs concernant le système de cote de la bibliothèque a eu des conséquences importantes sur le modèle de données et l'implémentation. Le modèle de données a dû être intégralement repris pour refléter ces nouvelles informations ; toute l'application serveur, ainsi que le script d'alimentation automatique (tous les deux déjà finalisés, voir le chapitre 6) ont dû subir des refontes complètes.

Les points essentiels de la gestion de projet ont bien été réalisés (analyse des besoins, conception/architecture, construction/développement, test/validation). Ils sont détaillés dans ce chapitre et les suivants. Cependant, compte tenu des points énoncés ci-dessus, aucune technique de génie logiciel (par ex. : pratiques Agile) n'a réellement pu être mise en œuvre dans ce travail, bien qu'une méthode en spirale (ou itérative) se soit naturellement appliquée, notamment pour le prototypage des interfaces utilisateurs.

Une fois les besoins clairement établis, les spécifications du projet ont été formalisées et validées par les professeurs encadrants. Elles sont disponibles en annexe sous la forme d'un cahier des charges (Annexe A.2).

Le système attendu est un outil scientifique centré sur le manuscrit. Il devra permettre la visualisation, l'annotation et la transcription de manuscrits disponibles sous la forme d'images en haute résolution.

Sans détailler tout l'historique de l'analyse des besoins, on se propose ici de synthétiser uniquement les spécifications qui en résultent.

2.2 Synthèse des spécifications

2.2.1 Visualisation

En terme de visualisation, le système devra :

1. Permettre une visualisation fluide et rapide des images en haute résolution (manuscrits) via un navigateur web.
   - Un *viewer* accélère la vitesse de visualisation des images en ne téléchargeant que les *tiles*[1] nécessaires à l'affichage de la portion d'image.
2. Offrir deux vues :
   - La vue de **consultation** permettant de consulter le manuscrit, ses annotations et sa transcription. L'espace pour afficher l'image est maximisé.
   - La vue d'**édition** permettant de créer/modifier/supprimer des annotations et/ou des transcriptions. L'espace pour afficher l'image est réduit afin d'afficher un éditeur de texte.
3. Permettre d'afficher rapidement les feuillets liés au feuillet courant :
   - Une liste ou un ensemble de miniatures (*thumbnails*) permet de sélectionner un autre feuillet et de l'afficher.
4. Permettre d'afficher ou de masquer les zones d'annotations d'un manuscrit (rectangle sur l'image) selon le désir de l'utilisateur.
   - Les annotations peuvent être affichées comme des couches/calques (*layers*) ou selon des filtres (fonction limitée dans ce prototype).
     Par exemple : Afficher uniquement les annotations d'un utilisateur, afficher toutes les annotations, etc.

2.2.2 Édition d'annotations/transcriptions/métadonnées

En ce qui concerne la transcription du manuscrit et les annotations, le système devra :

1. Permettre l'annotation du manuscrit :
   - Une annotation est définie par :
     - une donnée, c'est-à-dire l'annotation en tant que telle (un texte, un lien...),
     - des métadonnées (date de création, auteur de l'annotation...), et
     - un lien à une zone, c'est-à-dire une portion/sélection du manuscrit.
   - L'annotation est liée au manuscrit (création) via une sélection sur l'image (par ex. : un rectangle).
   - L'annotation est formatée en HTML et saisie via un éditeur *WYSIWYG*[2].
2. Permettre la transcription du manuscrit :
   - Comme il peut y avoir plusieurs interprétations, plusieurs transcriptions peuvent exister pour un même manuscrit ou pour un ensemble de manuscrit (voir la section 4.1.2)
   - Une transcription est définie comme une séquence d'éléments de transcription (voir la section 4.1.2).
   - Un élément de transcription est défini par :
     - une donnée, c'est-à-dire la transcription textuelle d'une portion du manuscrit,
     - des métadonnées (date de création, auteur de la transcription...), et
     - un lien à une zone, c'est-à-dire la portion du manuscrit correspondant à la transcription textuelle.

1. Une *tile* (ou tuile) est une fraction d'une image - voir la section 3.5.2
2. `http://en.wikipedia.org/wiki/WYSIWYG`

- Un élément de transcription est formaté en HTML et saisi via un éditeur *WYSIWYG*.

3. Offrir aux utilisateurs un menu ou une barre d'outils pour pouvoir saisir leurs annotations de manière interactive. Les options du menu ou de la barre d'outils consistent à :
   - Délimiter graphiquement une zone sur le manuscrit avec un rectangle
   - Saisir une annotation pour la lier à la zone
   - Éditer une annotation existante en gardant l'historique des modifications

En ce qui concerne les métadonnées, le système devra :

4. Permettre le stockage des métadonnées du manuscrit :
   - Ces métadonnées sont liées au manuscrit et donnent des informations sur l'objet physique, plutôt que sur son contenu (le catalogue de la BGE et les annotations servent déjà ce but).
     Par exemple : cote, emplacement physique, condition de l'objet, etc.
   - L'information délivrée par les métadonnées est de moindre importance pour l'analyse du contenu du manuscrit. Elles ne sont donc pas affichées sur le site ou alors partiellement.
     Par exemple : Affichage uniquement de la cote pour faciliter les références.

2.2.3 Stockage

En matière de stockage des images :

1. Le système devra stocker efficacement des images en haute résolution (en moyenne entre 20MB à 60MB, et jusqu'à 101MB)
   - À la suite d'un traitement *offline*, les images sont découpées en *tiles* (voir section 3.5.2).
   - Les *tiles* doivent être accessibles via internet.
2. Chaque feuillet (image originale + *tiles*) est identifié par une URI [3], basé sur sa cote.
3. Pour prévenir des problèmes liés aux droits d'auteur et à la propriété intellectuelle, l'image originale n'est pas forcément accessible directement via internet. Seules les *tiles* sont accessibles librement. L'accès aux *tiles* utilise l'URI du manuscrit/feuillet pour identifier la ressource.

En matière de stockage de l'information :

4. Les ressources (annotations, transcriptions, métadonnées, etc.) doivent être stockées dans un triplestore et publiées sur le web sémantique (grâce à un *endpoint* SPARQL [4]).

2.2.4 Mise à jour du contenu du site

En matière de mise à jour du contenu du site (ajout de photographies de manuscrits) :

1. Une chaîne de traitement (*workflow*) doit permettre d'alimenter automatiquement le site. Le processus doit être suffisamment simple pour que son utilisation ne requière aucune compétence particulière.
   - Les fichiers sont déposés sur un serveur (par FTP, par exemple). Périodiquement, un script vérifie la présence de nouvelles images. Celles-ci sont traitées selon leur nom.
   - Le nommage des fichiers doit respecter une convention de nommage prédéfinie et reflétant la cote du manuscrit que l'image représente.
   - La cote est extraite du nom de fichier, ce qui permet l'ajout des métadonnées dans le triplestore.
   - Les images sont également converties dans un format permettant l'utilisation de *tiles*.

3. `http://fr.wikipedia.org/wiki/Uniform_Resource_Identifier`
4. `http://en.wikipedia.org/wiki/SPARQL`

2.2.5 Recherche d'information

En matière de recherche d'information :

1. Les utilisateurs souhaitent pouvoir faire des recherches textuelles (*full text*) dans les annotations et les transcriptions.
   - Le texte des annotations/transcriptions est formaté en HTML. Les utilisateurs veulent pouvoir faire des recherches textuelles sur le contenu ; la mise en forme HTML ne fait pas partie des besoins de recherche (par ex. : recherche de mots en gras, etc.).
2. Les utilisateurs souhaitent également pouvoir faire des recherches textuelles (*full text*) sur la cote.

2.2.6 Navigation

En matière de navigation :

1. Les utilisateurs souhaitent pouvoir trouver des manuscrits en naviguant dans une structure la plus proche possible de celle des archives de la bibliothèque.
   - La structure des archives de la bibliothèque est directement obtenue depuis les métadonnées extraites de la cote.

Chapitre 3

État de l'art

Ce projet se place dans les *digital humanities*. Par conséquent, on commence, dans cet état de l'art, par explorer les systèmes similaires qui existent dans ce domaine et, comme on le verra, aucun ne correspond exactement à nos besoins. Cela nous amène à considérer les formats d'annotation et les formats de transcription, la manière de stocker l'information dans un triplestore, puis les techniques d'affichage d'images en haute résolution, ainsi que les formats desdites images.

Bien que cet état de l'art prenne en compte toutes les technologies et tous les logiciels, de manière générale on privilégie l'utilisation de logiciels libres (*open source*). Dans la mesure du possible, on souhaitera également utiliser des solutions offrant plus de flexibilité en termes de technologies ou de protocoles. Ces choix se justifient par le fait que le projet doit être extensible, adaptable et ouvert à une évolution future. Cette thèse propose une "première brique" à cet édifice et doit, par conséquent, envisager les possibilités d'évolution le plus largement possible.

Les logiciels *open source* permettent de ne pas dépendre d'un fournisseur, de sa stratégie et de ses potentielles modifications de licences (Google par exemple) ; ils nous permettent de pouvoir modifier le code à notre guise et d'en rester maîtres. Cette flexibilité est un aspect important pour l'indépendance et la pérennité de ce projet. Il faut tout de même faire attention au choix des logiciels *open source* que l'on décide d'utiliser, car certains sont moins avancés ou moins fiables que leur équivalent payant. Il arrive aussi que le développement de certains soit abandonné.

3.1 Projets *digital humanities* similaires

Il y a un grand intérêt dans le domaine des *digital humanities* pour la consultation d'archives numérisées. Beaucoup d'universités, de bibliothèques ou de musées proposent la visualisation de ces objets uniques sous la forme de photos hautes résolutions. Certains offrent aussi des descriptions annexes (métadonnées, catalogue...).

Il existe un nombre très important de projets et quelques études sur la question. Cette section passe en revue les systèmes existants ou ayant un intérêt pour l'aboutissement de ce projet. On s'intéresse également aux méthodes utilisées par d'autres en matière d'affichage, d'annotation et de navigation. Vu l'ampleur de la tâche, il n'est simplement pas possible de couvrir l'ensemble de ce qui existe. Seule une sélection non exhaustive des projets les plus pertinents (ou les plus représentatifs) est présentée ici.

3.1.1 Quelques projets représentatifs du domaine

Schopenhauer source [1] présente les manuscrits de Schopenhauer. Ils sont répertoriés dans un catalogue par "cahier"/registre [2]. Lorsqu'on visualise un manuscrit, la navigation par miniature se trouve à gauche et les pages sont affichées en vis-à-vis. Il est possible de zoomer sur l'image, de la télécharger ou de l'imprimer. Un viewer Flash non identifié permet la rotation et le zoom, mais il ne semble pas utiliser un système de *tiles*.

ITEM (l'Institut des TExtes et Manuscrits modernes) est une Unité Mixte de Recherche CNRS / ENS qui se consacre à l'étude des manuscrits d'écrivains pour élucider les processus de la genèse d'une œuvre littéraire [3]. ITEM propose notamment une plateforme collaborative pour partager des ressources et mettre des fichiers (images, sons, etc.) à disposition des autres membres d'un projet, ainsi que la présentation publique de résultats de recherche. Rien ne laisse penser qu'ITEM propose des archives d'images en haute résolution, ni un système de visualisation.

Old Maps Online [4] est une ressource extrêmement riche pour la mise en ligne de cartes anciennes. Outre la visualisation de cartes, le site contient notamment des conseils pour numériser et publier en ligne des images TIFF ou JPEG2000. Les interfaces proposées sont axées pour des données cartographiques, et ne sont pas adaptées aux besoins de ce projet. Cependant, la documentation sur les technologies (formats d'images, viewer, serveurs d'image, etc.) est très bien réalisée et très complète.

Ohio Digital Resource Commons [5] détaille toutes les données de leurs manuscrits en ligne (titre, auteur, description, etc.). Un viewer IIPZoom (Flash) est intégré dans la page [6]. Une liste de *thumbnails* des photos associées se trouve en bas du viewer et permet de faire défiler les images dans le viewer (navigation). En fin de page se trouve la liste des images associées et chaque image originale peut être téléchargée (en JPEG2000). Le viewer est trop petit pour permettre une lecture agréable du manuscrit, mais il est possible de le passer en plein écran.

The National Library of Australia met à disposition le journal de James Cook [7]. Le site propose deux versions des images (*View* pour une vue plus large, et *Examination* pour une vue plus précise). Il y a aussi l'*Interactive Image* qui permet de zoomer avec une fluidité laissant à désirer (la page est rechargée à chaque zoom/*pan* [8]).

Le **Musée d'Israël** de Jérusalem offre la visualisation de cinq des *Manuscrits de la mer Morte* numérisés : Digital Dead Sea Scrolls [9]. L'interface pour la visualisation des (fragments de) rouleaux a été développée en partenariat avec Google (et se base probablement partiellement sur Google Maps). Elle permet de zoomer (système de *tiles*), de naviguer entre les différents fragments d'un rouleau (présentation d'un rouleau miniature avec des numéros de fragments) ; des traductions de portions de texte se trouvent sur les fragments (zones de texte cliquables et *popup* de la traduction anglaise).

The Lancelot-Graal Project [10] est un projet, *a priori* intéressant, pour la mise en ligne des manuscrits médiévaux. Leur approche est d'utiliser un système de GIS pour lier de l'information

1. http://www.schopenhauersource.org/
2. http://www.schopenhauersource.org/type_list.php?type=manuscript
3. http://www.item.ens.fr/
4. http://help.oldmapsonline.org/
5. http://drc.ohiolink.edu/
6. http://drc.ohiolink.edu/handle/2374.OX/62434
7. http://nla.gov.au/nla.ms-ms1
8. http://hea-www.harvard.edu/RD/saotng/panzoom.html
9. http://dss.collections.imj.org.il/
10. http://www.lancelot-project.pitt.edu/lancelot-project.html

à l'image. Des zones dessinées sur l'image sont cliquables et ouvrent une nouvelle page contenant des portions du manuscrit (qu'ils nomment *Key Passages*) et leurs transcriptions. Le site est très basique. Bien qu'il semble y avoir du contenu lié au manuscrit, l'interactivité avec l'image est nulle (pas de zoom, etc.).

Le projet ne semble pas être ouvert (ni *open source*, ni même payant), et il n'est pas non plus très clair s'il est encore d'actualité ou non.

The National Gallery de Londres permet la visualisation d'œuvres de Raphaël : Raphael Research Resource [11]. Un viewer IIPMooViewer modifié présente des peintures disposées en grille, permettant de voir un aperçu des œuvres et de zoomer dessus (toutes les images sont dans le même viewer). Ce système est couplé à une base de données et la disposition en grille est générée en forçant des paramètres du viewer, en fonction du nombre d'images fournies et de leurs dimensions. En cliquant sur une photo, des informations sont affichées sur la gauche de la page (comme le titre, l'artiste, le type de peinture et de support utilisés, etc.). Un viewer fonctionnant sur le même principe (toutes les images dans le même viewer avec des métadonnées correspondant à l'œuvre sélectionnée) présente une partie de la collection du musée [12].

Ces interfaces, bien que très utiles pour montrer plusieurs images en une fois, autorisent l'utilisateur à zoomer entre les images, sur des zones ne contenant rien et de facilement se perdre. Une vue permettant de voir *Images & Texts* est disponible pour *Raphael Research Resource*, mais aucun texte n'est affiché.

The National Gallery de Londres propose encore des pages de démonstration (ou des preuves de concepts) avec des versions modifiées du viewer IIPMooViewer [13]. L'un des viewers [14] intègre une navigation par *thumbnail* dans la partie inférieure du viewer (en *filmstrip*) et des boutons permettant de cacher ou d'afficher certains éléments de la page (*thumbnails*, mini-map).

3.1.2 Nietzsche Source

Le site *Nietzsche Source* [15] est divisé en deux parties :

- La version numérique de l'édition critique allemande de l'œuvre de Nietzsche établie par Giorgio Colli et Mazzino Montinari.
- La reproduction numérique en fac-similé du corpus nietzschéen incluant les premières éditions de ses œuvres, ses manuscrits, sa correspondance ainsi que les documents biographiques le concernant. Plus de neuf mille pages sont actuellement disponibles.

La version numérique de l'édition critique est une version textuelle avec un affichage conçu pour ressembler à une édition papier. Un menu sur la gauche permet de naviguer entre les éditions ou à l'intérieur d'une édition.

La reproduction numérique propose deux sous-sections : les *œuvres* et les *manuscrits*.
Les *œuvres* sont essentiellement des numérisations de textes imprimés, et les *manuscrits* sont des numérisations de cahiers, d'épreuves annotées, de carnets, etc.

La visualisation se fait avec le viewer IIPZoom (Flash). Un menu à gauche permet la navigation dans l'objet courant. Chaque feuillet peut être affiché individuellement et on passe au suivant avec des flèches (dans le bord de l'image) ou avec le menu de navigation. Il est aussi possible d'afficher les feuillets côte à côte, pour donner l'impression de voir un livre ouvert

11. `http://cima.ng-london.org.uk/documentation/`
12. `http://cima.ng-london.org.uk/collection/index.php`
13. `http://research.ng-london.org.uk/wiki/index.php/National_Gallery_IIPImage`
14. `http://cima.ng-london.org.uk/iip/rembrandt/`
15. `http://www.nietzschesource.org/`

(N.B. : dans ce cas deux viewers sont utilisés, soit un par image). Le viewer Flash propose des options de rotation, zoom, et quelques effets (contraste, luminosité, etc.).

Nietzsche Source est souvent cité comme l'un des sites les mieux réalisés du domaine, notamment grâce à la mise en page de l'édition critique. On peut tout de même noter qu'il n'y a pas de liens entre les éditions critiques et les reproductions numériques.

3.1.3 Utrecht University Library Special Collections

From scan to delivery : Special Collections and IIPImage at Utrecht University Library[16] est un article invité du blog d'IIPImage, écrit par des membres de l'Université d'Utrecht. Celui-ci décrit le *workflow* de numérisation et le choix des technologies de leur projet.

L'Université d'Utrecht utilise le viewer "Open Library Bookreader". Lorsque l'utilisateur veut voir une meilleure image, il peut utiliser le viewer IIPZoom (si Flash est disponible), ou IIPMooViewer et les images (format JPEG2000) sont servies avec le serveur IIPImage.

3.1.4 Wellcome Digital Library

Le blog *Wellcome Digital Library*[17] poste, notamment, des articles très bien rédigés sur leur stratégie de numérisation. D'après l'un des articles[18], ils utilisent IIPImage pour servir dynamiquement des *tiles* au format Deep Zoom depuis des fichiers JPEG2000. Les *tiles* sont affichées avec le viewer Seadragon.

Toutes les métadonnées sont stockées dans une structure METS à laquelle l'application accède via une API publique, utilisant un "format de données intermédiaire" développé par leurs soins. Cela leur permet de retrouver, par exemple, toutes les pages d'un livre et leur ordre.

L'un des portails est dédié aux manuscrits ayant trait à l'histoire de la médecine arabe : Wellcome Arabic Manuscripts Online[19]. La figure 3.1 montre l'interface utilisateur. Le panneau de gauche affiche les métadonnées, tandis que celui de droite présente l'image avec un viewer Seadragon. À gauche de l'image se trouve une navigation par miniatures (*thumbnails*). Un menu déroulant au-dessus de l'image permet d'exporter l'image ou certaines métadonnées dans des formats prédéfinis.

3.1.5 Bentham Project

The Bentham Project[21] est un projet visant à transcrire les manuscrits du philosophe et juriste Jeremy Bentham. L'approche se base sur du *crowdsourcing*[22], où n'importe qui peut contribuer à la transcription. Les transcriptions sont ensuite validées par des experts. À la fin du projet, il est question de rendre ces outils accessibles à d'autres projets[23].

16. `http://iipimage.sourceforge.net/2012/05/from-scan-to-delivery-special-collections-and-iipimage-at-utrecht-university-library/`
17. `http://wellcomedigitallibrary.blogspot.co.uk/`
18. `http://wellcomedigitallibrary.blogspot.co.uk/2012/05/developing-player-for-wellcome-digital.html`
19. `http://wamcp.bibalex.org/home`
20. capture d'écran de `http://wellcomelibrary.org/about-us/projects/digitisation/`
21. `http://www.ucl.ac.uk/Bentham-Project`
22. `http://en.wikipedia.org/wiki/Crowdsourcing`
23. `http://liber.library.uu.nl/index.php/lq/article/view/URN:NBN:NL:UI:10-1-113603`

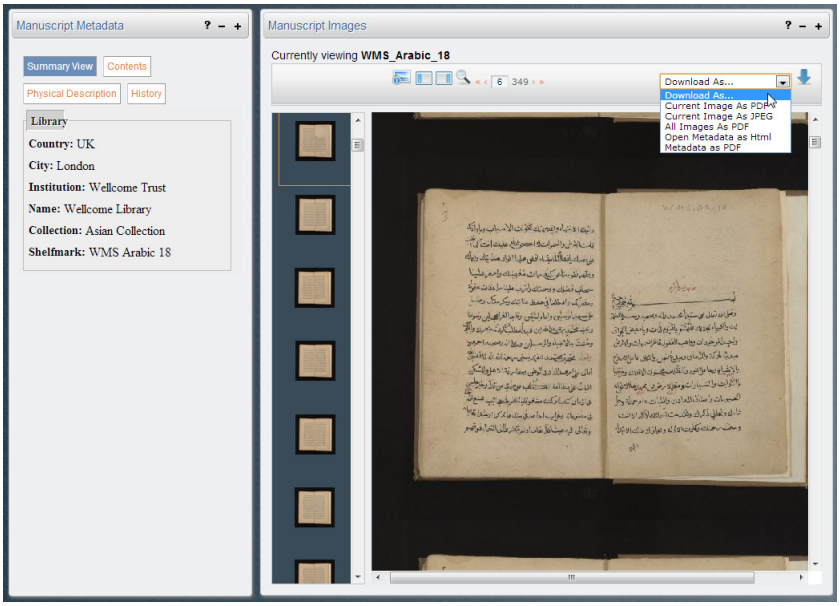


FIGURE 3.1 – *Wellcome Arabic Manuscripts Online viewer* [20]

Le *Transcription Desk* permet de voir la progression des transcriptions des manuscrits. Lors de la transcription, un éditeur de texte se trouve sur la gauche, et sur la droite se trouve un viewer avec l'image. Le viewer est Zoomify Express pour la version Flash, ou Brainmaps Javascript zoomviewer [24] pour la version Javascript. Les deux viewers utilisent les *tiles* au format Zoomify.

Les transcriptions sont encodées avec TEI et toute l'interface du site est basée sur MediaWiki avec un *skin* modifié [25].

3.1.6 Bilan intermédiaire

Aucun des systèmes décrits dans cet état de l'art ne correspond exactement à nos besoins. Certes, il existe des projets avec une volonté partiellement similaire, mais rien qui ne soit directement réutilisable.

La plateforme qui s'approche le plus de nos objectifs est *Nietzsche Source*, cependant le système ne permet aucune interaction des utilisateurs sur les images ou les textes. Ce n'est pas un outil de travail ou une plateforme destinée à la collaboration.

Bentham Project propose une approche intéressante pour la transcription du corpus via du *crowdsourcing*. L'interface et le code ne sont malheureusement pas ouverts et dans tous les cas, des adaptations devraient être faites pour les annotations.

Tous les autres projets mentionnés dans cet état de l'art servent à la consultation d'images. Certains offrent différentes formes de navigation, ou l'affichage simultané de métadonnées ou encore des subtilités au niveau de l'affichage.

Bentham Project et T-Pen (que l'on décrira dans la section 3.4.4.2) sont les deux seuls projets à s'imposer comme des outils de travail originaux en proposant aux utilisateurs une interaction avec l'image du manuscrit.

La suite de cet état de l'art explore la manière de représenter des annotations et des transcriptions, ainsi que le moyen de les stocker, avant d'étudier la problématique de l'affichage des images en haute résolution et du format des images.

24. `http://brainmaps.org/index.php?p=brain-maps-api`
25. `http://www.transcribe-bentham.da.ulcc.ac.uk/td/Technical_Requirements`

3.2 Format des annotations/transcriptions et catalogage

Pour représenter les données (annotations et transcriptions), il faut choisir un modèle pour structurer l'information d'une façon claire et répondant aux besoins des utilisateurs, notamment en matière de recherche d'information. *De facto*, deux formats XML sont recommandés pour la représentation et la transcription de manuscrits : METS et TEI. Notons aussi qu'il existe des standards/recommandations pour les métadonnées [26] (Dublin Core, MARC,...) qui ne sont pas étudiés ici.

3.2.1 METS

METS (Metadata Encoding and Transmission Standard) [27] est un standard pour la description complète d'objets numériques.

> *"The METS schema is a standard for encoding descriptive, administrative, and structural metadata regarding objects within a digital library, expressed using the XML schema language of the World Wide Web Consortium. "* [28]

METS permet essentiellement de gérer les métadonnées pour une bibliothèque d'objets numériques. METS n'est pas conçu pour contenir une version encodée de l'objet :

Extrait du document "METS : vue d'ensemble & tutoriel", partie "Section des fichiers" [29] :

> *"La section des fichiers (<fileSec>) comprend un ou plusieurs éléments <fileGrp> qui servent à regrouper des fichiers de même nature. Un <fileGrp> liste tous les fichiers constituant une version électronique distincte de l'objet de bibliothèque numérique. Par exemple, on pourrait avoir des éléments <fileGrp> distincts pour les vignettes, les masters des images, les versions PDF, les versions textuelles encodées en TEI, etc."*

Le format METS sert bel et bien à structurer des métadonnées et des documents, parmi lesquels il est possible de lier les "versions textuelles encodées en TEI" (par exemple), mais METS n'est pas lui-même adapté pour l'encodage du texte.

3.2.2 TEI Recommandations

TEI (Text Encoding Initiative) [30] est un consortium, à l'instar du W3C pour le web, qui propose des recommandations afin de normaliser l'encodage de toutes sortes de documents sous forme numérique.

Ces recommandations visent donc à formaliser la transcription d'un manuscrit et de métadonnées liées à celui-ci (l'état physique du manuscrit, etc.).

Dans notre cas, on s'intéressera à la dernière version des recommandations, à savoir la version P5 (*P5 Guideline*) [31], plus précisément les chapitres 10 et 11, respectivement *Manuscript Description* et *Representation of Primary Sources*.

26. `http://help.oldmapsonline.org/metadata`
27. `http://www.loc.gov/standards/mets/`
28. `http://www.loc.gov/standards/mets/mets-home.html`
29. `http://www.loc.gov/standards/mets/METSOverview.v2_fr.html#filegrp`
30. `http://www.tei-c.org`
31. `http://www.tei-c.org/release/doc/tei-p5-doc/en/Guidelines.pdf`

3.2.2.1 Manuscript Description

> *"This module defines a special purpose element which can be used to provide detailed descriptive information about handwritten primary sources. Although originally developed to meet the needs of cataloguers and scholars working with medieval manuscripts in the European tradition, the scheme presented here is general enough that it can also be extended to other traditions and materials, and is potentially useful for any kind of inscribed artefact.*
> *The scheme described here is also intended to accommodate the needs of many different classes of encoders. On the one hand, encoders may be engaged in retrospective conversion of existing detailed descriptions and catalogues into machine tractable form ; on the other, they may be engaged in cataloguing ex nihilo, that is, creating new detailed descriptions for materials never before catalogued. "*

3.2.2.2 Representation of Primary Sources

> *" This chapter defines a module intended for use in the representation of primary sources, such as manuscripts or other written materials. Section 11.1. Digital Facsimiles provides elements for the encoding of digital facsimiles or images of such materials, while the remainder of the chapter discusses ways of encoding detailed transcriptions of such materials. [...] Detailed metadata relating to primary sources of any kind may be recorded using the elements defined by the manuscript description module discussed in chapter 10. Manuscript Description, but again the present module may be used independently if such data is not required. "*

Le format TEI permet donc d'encoder la transcription du manuscrit en XML ; il sert également, au besoin, à la description de l'objet physique.

3.2.3 Bilan intermédiaire

Les formats METS et TEI, utilisés conjointement, permettent de structurer tous les documents relatifs aux manuscrits et d'utiliser des fichiers formatés TEI dans la liste de METS pour la partie transcription.

De là émergent trois problèmes. Premièrement, nous souhaitons pouvoir définir plusieurs transcriptions par manuscrit et il n'est pas clair si cela est possible. Deuxièmement, qu'il s'agisse de transcriptions ou d'annotations, elles doivent être liées à des zones sur des images, ce que ces formats n'offrent pas directement. Troisièmement, TEI est prévu pour les transcriptions, mais pas pour les annotations.
Ces deux formats ne correspondent pas tout à fait à ce projet, car ils sont destinés à des fins d'archivage ou de catalogage, alors que nous souhaitons les utiliser avec un outil de recherche dynamique et flexible.

De manière générale, ces formats présentent l'avantage d'être facilement exportables vers des systèmes compatibles. Leurs désavantages principaux résident dans le fait qu'ils sont très lourds à mettre en place et qu'il faut s'adapter à la structure fournie. Cela peut devenir particulièrement problématique lors d'une mise à jour ou d'une révision des spécifications, car cela implique la conversion de toutes les données pour maintenir la compatibilité.

En pratique, une alternative assez répandue consiste à utiliser une structure interne au projet et à être capable d'en extraire de l'information au format TEI ou METS. Ici, on souhaite un modèle permettant de faciliter les fonctions suivantes :

- Afficher les annotations liées à un manuscrit
- Afficher la transcription d'un manuscrit
- Ajouter une annotation ou une transcription à un manuscrit
- Modifier une annotation ou une transcription liée à un manuscrit
- Supprimer une annotation ou une transcription liée à un manuscrit
- Procéder à la recherche d'information sur un manuscrit ou sur le corpus

La solution retenue dans ce projet est d'utiliser du texte formaté en HTML pour les annotations et les transcriptions. Ce texte sera stocké au sein d'un modèle au format RDF (voir le chapitre 4). Ce modèle pourra permettre, au besoin, d'extraire une information formatée METS ou TEI.

3.3 Triplestore / Quadstore

3.3.1 Stockage des métadonnées et des annotations

Il y a deux méthodes pour stocker des annotations : soit directement dans le fichier image (dans les métadonnées), soit séparément. Certains formats d'images permettent de stocker des métadonnées dans l'entête du fichier (EXIF [32]). Il est par exemple possible de stocker des annotations directement dans un fichier TIFF :

> *"[TIFF] permet le stockage d'image par bloc (on parle alors de TIFF tuilé), et aussi de multiples images par fichier, des images alternatives en basse résolution, des annotations sous forme de courbes et de texte, etc."* [33]

L'intérêt de stocker des métadonnées dans le fichier TIFF est qu'elles y sont liées (tout dans un fichier). Il semble néanmoins préférable d'utiliser un système de stockage de l'information séparé de l'image, afin d'éviter d'écrire directement dans le fichier image (rapidité de lecture, risque de corruption à l'écriture, gestion des accès concurrents...). De plus, on ne sait pas à l'avance quelle quantité de données peuvent représenter les annotations ; le fait de tout écrire dans un seul fichier comporte des risques. Le fichier pourrait grandir indéfiniment, rendant sa manipulation de plus en plus lourde. La sauvegarde de nouvelles annotations pourrait s'avérer impossible si l'entête du fichier (dans laquelle sont les métadonnées) impose une limite de taille, ou si le fichier atteint la taille maximum permise par le système de fichiers [34].

Rappelons qu'à terme, le projet doit permettre l'indexation et l'analyse des transcriptions et des annotations. Afin de faciliter leur manipulation, il semble plus que naturel que ces données textuelles soient stockées dans un format courant pour ce genre de tâche (XML, base de données, RDF), plutôt que dans l'entête d'un fichier image.

Ici, on préfère utiliser un stockage séparé de l'image. Si les annotations et les transcriptions sont dans des fichiers (fichier texte, XML), une lecture du ou des fichiers concernés doit être faite à chaque lecture/écriture (lent). Si elles sont stockées dans un système optimisé (base de données XML native, base de données relationnelle, triplestore RDF...), les accès sont plus rapides.

Les bases de données relationnelles (MySQL, PostgreSQL...) sont très souvent utilisées, alors qu'il existe des alternatives. Certains moteurs de Wiki, comme Mediawiki par exemple, offrent des possibilités d'accéder aux données sans passer par l'interface web. Outre les *skins* pour

32. `http://en.wikipedia.org/wiki/Exchangeable_image_file_format`
33. `http://fr.wikipedia.org/wiki/Tagged_Image_File_Format`
34. Par exemple : limite de la taille des fichiers à 4GB avec FAT32

modifier la mise en page, il existe une API qui permet notamment de faire des requêtes sur les données [35] et même d'éditer (ou de créer) des pages [36]. En utilisant certains paramètres (GET), il est aussi possible d'afficher simplement une page [37] sans une bonne partie du formatage (et des menus) [38].

Le cahier des charges impose l'utilisation d'un modèle de données pour le web sémantique, plutôt qu'une base de données relationnelle. On utilise donc un triplestore avec des données RDF.

3.3.2 RDF/SPARQL

Le web sémantique est basé sur Resource Description Framework (RDF), qui est un modèle de graphe permettant une représentation souple, et qui présente l'avantage de pouvoir évoluer facilement et de rendre les données beaucoup plus accessibles. C'est une alternative aux bases de données relationnelles, bien que leurs fonctionnements ne soient pas comparables. Ce modèle de graphe est destiné à décrire de façon formelle les ressources web et leurs métadonnées, afin de permettre le traitement automatique de telles descriptions [39]. Un document structuré en RDF est un ensemble de triplets, où un triplet est une association :
(sujet, prédicat, objet)
Contrairement à une base de données ou de l'XML, où le modèle de données doit être prédéfini avant d'y insérer des données, la structure en graphe de RDF permet d'avoir un modèle évolutif en fonction des besoins.

Dans le cadre de ce travail, nous souhaitons pouvoir écrire des requêtes pour rechercher des informations, mais également en ajouter, en modifier et en supprimer.
Les langages **SPARQL** et **SPARQL Update** servent ce but :

- SPARQL (SPARQL Protocol And RDF Query Language) [40] [41] - pour la recherche :
  - lire/chercher des données (SELECT / CONSTRUCT / ASK / DESCRIBE)
- SPARQL Update [42] - pour la manipulation
  - ajouter des données (INSERT)
  - supprimer des données (DELETE)
  - modifier des données (DELETE/INSERT)

N.B. : Des exemples de requêtes SPARQL / SPARUL se trouvent en annexe (voir Annexe B.4).

Le langage **SPARQL Update** [43] (parfois, SPARUL pour **SPAR**QL **U**pdate **L**anguage) permet d'insérer, de supprimer et de modifier (mettre à jour) des données RDF dans un triplestore, en utilisant une syntaxe dérivée de SPARQL. La recommandation du W3C pour SPARQL Update est en cours d'écriture. Le *Working Draft* de la recommandation est disponible et certains triplestores, dont OpenRDF Sesame, l'implémentent déjà en l'état. Attention cependant, car les documentations trouvées sur internet se réfèrent parfois à d'anciennes versions du *draft* avec des syntaxes de requêtes différentes ou abandonnées. La plus notable est la requête `MODIFY` qui n'existe plus, car remplacée par `DELETE/INSERT` [44].

35. `http://www.mediawiki.org/wiki/API:Query`
36. `http://www.mediawiki.org/wiki/API:Edit`
37. `http://en.wikipedia.org/w/index.php/Albert_Einstein`
38. `http://en.wikipedia.org/w/index.php?title=Albert_Einstein&action=render`
39. `http://fr.wikipedia.org/wiki/Resource_Description_Framework`
40. `http://www.w3.org/TR/sparql11-protocol/`
41. `http://www.w3.org/TR/sparql11-query/`
42. `http://www.w3.org/TR/sparql11-update/`
43. `http://en.wikipedia.org/wiki/SPARUL`
44. `http://www.w3.org/Submission/SPARQL-Update/`

3.3.3 Triplestore et *endpoint* SPARQL

Un triplestore est une base de données particulière permettant le stockage, la récupération et la manipulation de données RDF, notamment grâce au langage de requête SPARQL.

Dans un triplestore, l'information y est stockée sous forme de triplets [45], c'est-à-dire un graphe composé du **sujet**, du **prédicat** et de l'**objet**. Un quadstore ajoute un quatrième attribut : le **nom du graphe**. Dans ce document, on ne fait pas la distinction entre triplestore et quadstore.

L'utilisation d'une ontologie pour stocker les annotations et les transcriptions nécessite de pouvoir accéder au triplestore depuis l'application web. En effet, l'accès au triplestore peut se faire de différentes façons. Heureusement Sesame offre plusieurs API, dont une API HTTP/REST permettant l'envoi de requêtes par paquets HTTP ; on parle alors d'***endpoint* SPARQL**.

3.3.3.1 Choix du triplestore

En annexe se trouve une liste des triplestores et autres librairies compatibles avec le langage de requête SPARQL (Annexe C.1.1).

Il existe une quantité importante de triplestores [46]. Seule une sélection a été testée :

- OpenRDF Sesame [47]
- AllegroGraph [48]
- OpenLink Virtuoso Universal Server [49]

N.B. : Procédures d'installation en annexe (Annexe B.3)

Bien que ces trois produits correspondent à nos besoins, le choix s'est arrêté sur OpenRDF Sesame (*open source*, stable, bonne réputation...) [50]. Il supporte aussi les versions les plus récentes des recommandations du W3C [51]. Si d'aventure Sesame ne convenait pas sur le long terme, le transfert de Sesame à un autre triplestore est possible en exportant les données en RDF/XML.

3.3.3.2 Moteur de recherche / Indexation / Full text search

L'un des objectifs de ce travail est de pouvoir faire des recherches textuelles sur les données stockées dans le triplestore. Dès lors, la recherche peut être effectuée selon différentes approches :

- Utilisation d'un système d'indexation
  - Certains triplestores permettent nativement la recherche *full text* (Allegrograph, BigOWLIM) [52] .
  - Utilisation de librairies tierces pour réaliser l'indexation sur le triplestore.

45. `http://fr.wikipedia.org/wiki/Triplestore`
46. `http://en.wikipedia.org/wiki/Triplestore#List_of_implementations`
47. `http://openrdf.org/`
48. `http://www.franz.com/`
49. `http://virtuoso.openlinksw.com/`
50. Dans ce document, on se réfère indifféremment à *OpenRDF Sesame*, ou simplement *Sesame*
51. `http://rivuli-development.com/2011/08/sesame-2-5-0-released-sparql-1-1-queryupdate-support-binary-rdf/`
52. `http://answers.semanticweb.com/questions/8676/do-you-use-full-text-search-with-sparql-if-so-how-and-why`

  - uSeekM IndexingSail - basé ElasticSearchIndexer (Lucene *wrapper*). "*It can be used with any RDF database (triplestore) with an OpenRdf Sail layer.*"
  - LuceneSail [53] est basé sur Lucene, mais pas de mise à jour depuis 2009.
  - SIREn [54] (un *plugin* pour Lucene) - d'après Renaud Delbru (chef du projet SIREn) semble meilleur pour la recherche par mots-clefs, mais pas forcément pour des requêtes SPARQL.

    > *"Compared to SIREn, Jena LARQ and Sesame Lucene Sail support fully SPARQL (graph-based query), while SIREn is restricted to simpler query (star-shaped query, and/or simple short RDF paths"* [55]

  - Tuqs [56] [57] est une API OpenRDF SAIL basée sur Lucene.
- Utilisation d'un *hack* qui extrait les données textuelles du triplestore pour les mettre dans une base MySQL pour profiter du *fulltext search engine* inclus dans la base de données [58].
- Approche simple : un moteur de recherche simple basé sur des requêtes SPARQL contenant un filtre par regex.
  Avantages : résultats toujours à jour,
  Désavantages : la recherche n'est pas optimisée ; aucun problème de performance pour une petite ontologie, mais pourrait poser des problèmes sur des jeux de données plus importants.

L'état de l'art semble indiquer que les solutions les plus efficaces pour le *full text* reposent plus ou moins toutes sur Lucene (*wrapper*, *plugin*...). Certains triplestores, comme Virtuoso ou Allegrograph, offrent aussi des possibilités (mais ça n'est pas clair sur quoi elles reposent).

D'autres liens, discussions, articles et *benchmarks* sont disponibles en annexe (Annexe C.1.2).

La solution adoptée dans ce travail est celle d'un moteur de recherche basé sur des requêtes SPARQL avec un filtre par regex. Cette solution est la moins intrusive tout en restant fonctionnelle. Tant que le jeu de données RDF n'est pas trop grand, elle permet d'avoir des résultats rapidement et toujours à jour. Le fonctionnement de ce moteur de recherche est décrit dans la section 6.5.2.

3.4 Affichage d'images en haute résolution

L'affichage d'images en haute résolution est une problématique centrale dans ce travail. Le format de l'image a également une importance capitale, car c'est lui qui permet la fluidité de l'affichage dans un navigateur web (voir la section 3.5). En effet, les versions numérisées des manuscrits de Saussure sont des photographies pouvant faire jusqu'à 100MB. Pour des raisons d'ergonomie et de temps d'accès, il n'est pas concevable de systématiquement servir à l'utilisateur une image complète : télécharger des images de cette taille n'est pas pratique du tout pour l'utilisateur (latence) et ceci créerait une charge (réseau) inutile au niveau du serveur.

Il faut donc utiliser un système qui permette de stocker les photographies, et de rapidement pouvoir visualiser l'image entière ou une portion de celle-ci, mais sans systématiquement la charger intégralement. Typiquement, on souhaite pouvoir agrandir/zoomer sur la zone d'intérêt en ayant une meilleure qualité d'image au fur et à mesure du zoom.

53. http://dev.nepomuk.semanticdesktop.org/wiki/LuceneSail
54. http://siren.sindice.com/
55. http://lists.deri.org/pipermail/siren/2010-January/000013.html
56. http://sourceforge.net/p/tuqs/wiki/Home/
57. http://www.turnguard.com/turnguard
58. http://www.openrdf.org/forum/mvnforum/viewthread?thread=1050

Pour ce faire une application serveur envoie des portions d'images (*tiles*) à une application client (viewer) dont le rôle est de les afficher correctement. Le serveur doit utiliser des images dans un format bien particulier (les formats sont décrits à la section 3.5). On considère aussi ici les possibilités en matière d'annotation, car si elles existent, elles sont généralement intégrées dans le viewer.

3.4.1 GIS (*Geographic information system*)

Les systèmes d'information géographique [59] (SIG ou souvent GIS pour *Geographic information system* en anglais) permettent la visualisation de cartes géographiques. Ils sont généralement constitués d'un logiciel serveur qui sert les cartes, d'une ou plusieurs bases de données et d'un logiciel client (ou d'une API permettant à un logiciel client d'accéder aux données).

La plupart des GIS peuvent afficher soit des cartes vectorielles, soit des images *raster* [60] [61]. Dans le cas de ce projet, les images sources sont des *raster images*.

Les systèmes GIS semblent être assez proches de ce que nous cherchons à réaliser et le sujet est suffisamment vaste pour ne pas l'ignorer. Il existe un nombre important de GIS. Seule une sélection non exhaustive est présentée ici.

3.4.1.1 Google Maps (with Google Maps API)

Le GIS le plus connu du grand public est sans doute Google Maps. Google permet de facilement utiliser l'API de GoogleMaps pour l'intégrer à n'importe quel site ou même à des applications mobiles. En effet, il est possible de charger une image autre qu'une carte [62] et profiter des fonctionnalités de Google Maps, telles que le zoom et l'annotation de points (points d'intérêts) ou de zones [63].

Google Maps présente des inconvénients notables : bien que gratuit, il n'est pas possible d'utiliser un serveur privé pour stocker les images/cartes. Cela implique que le projet est donc dépendant de Google, de ses conditions d'utilisation et de toutes les modifications qui peuvent être apportées au projet Google Maps (comme la limite de l'utilisation quotidienne introduite en 2011 [64]). En outre, les *tiles* doivent être générées dans un format bien précis (voir 3.5.4.4).

3.4.1.2 Esri ArcGIS

ArcGIS est un outil de référence pour les géographes. Il est très abouti et permet un nombre d'options important pour utiliser des cartes, y ajouter de l'information, etc. C'est un logiciel essentiellement graphique et une personnalisation peut se faire en développant des modules.

ArcGIS a plusieurs désavantages : premièrement, il est payant. Deuxièmement, il est orienté

59. http://fr.wikipedia.org/wiki/Système_d'information_géographique
60. http://gislounge.com/geodatabases-explored-vector-and-raster-data/
61. http://gis.stackexchange.com/questions/7077/what-are-raster-and-vector-data-in-gis-and-when-to-use
62. Create Zoomable Images Using The Google Maps API : http://blog.mikecouturier.com/2011/07/create-zoomable-images-using-google.html
63. How to Make and Annotate Maps with Google My Maps : http://www.tucows.com/article/1567
    Exemple ludique avec une carte d'un monde fictif de Tolkien - Lord Of The Rings Map : http://www.maplib.net/map.php?id=13#
64. http://googlegeodevelopers.blogspot.com/2011/10/introduction-of-usage-limits-to-maps.html

cloud et il n'est pas possible d'utiliser un serveur privé pour les images/cartes. Finalement, la version ArcGIS Online developer n'est pas encore disponible au moment de l'écriture de cet état de l'art.

3.4.1.3 Old Maps Online

Le site Old Maps Online n'est pas une solution en tant que telle, mais une ressource extrêmement riche pour la mise en ligne de cartes anciennes. Il contient une section dédiée aux outils de géoréférencement [65].

3.4.2 Serveur d'images / Création de *tiles*

Pour permettre un affichage fluide, l'image est affichée partiellement : seules des portions de l'image (*tiles*) sont affichées au fur et à mesure des requêtes (selon la position et le niveau de zoom). La section 3.5.2 explique ce mécanisme en détail.

La première solution est d'utiliser un ensemble de fichiers - des *tiles* statiques (Zoomify, Deep Zoom) - et seul un serveur web standard est requis. L'arborescence est stockée sur le serveur web et chaque fichier peut être accessible via de simples requêtes HTTP GET. La deuxième solution consiste à utiliser un fichier unique (PTIF, JPEG2000) et requiert d'utiliser un serveur web ainsi qu'un logiciel qui va permettre de servir les *tiles* (comme djatoka ou IIPImage, décrits respectivement dans les sections 3.4.2.3 et 3.4.2.4).

Dans les deux cas, un logiciel client qui s'exécute dans le navigateur web de l'utilisateur est nécessaire pour la visualisation et doit être compatible avec le protocole utilisé par le serveur (voir la section 3.4.3).

Cette section explore les différentes solutions pour servir des *tiles*. Il en existe beaucoup qui sont *open source* et gratuites (liste de logiciels ici [66]). Seules les plus pertinentes sont décrites ici.

3.4.2.1 Zoomify

Zoomify est une compagnie qui offre plusieurs solutions [67] pour l'affichage d'images en haute résolution sous forme de *tiles* au format JPEG ou PNG (voir section 3.5.4.4).

L'édition Zoomify Express [68] est gratuite et se compose d'un outil (Zoomify Converter) pour créer les *tiles*, ainsi que d'un lecteur Flash. Les fichiers générés peuvent simplement être placés sur un serveur web standard et l'image peut ensuite être visualisée avec le lecteur Flash fourni.

> ***"Is there any special server setup needed ?***
> *No. Zoomify Images can be viewed without any special server setup. Zoomify Images use just JPEGs, HTML, and a tiny Flash movie that serves as the 'viewer'. For this reason, they can be copied to any web server and immediately viewed by any site visitor without special server setup or special client software. "* [69]

65. `http://help.oldmapsonline.org/georeference`
66. `http://code.google.com/p/oldmapsonline/wiki/ImageServers`
67. `http://www.zoomify.com/compare.htm`
68. `http://www.zoomify.com/express.htm`
69. `http://www.zoomify.com/support.htm#a20081222_1435`

3.4.2.2 Deep Zoom

Deep Zoom est un logiciel propriétaire de Microsoft qui permet la visualisation d'images en haute définition. Bien que Deep Zoom soit propriétaire, il est tout à fait possible d'utiliser gratuitement le format *.dzi (voir section 3.5.4.4). L'arborescence contenant les *tiles* est simplement placée sur un serveur web, à l'instar de Zoomify. Le serveur ne nécessite donc pas de configuration particulière.

3.4.2.3 djatoka (aDORe djatoka)

djatoka est un serveur d'images *open source* écrit en Java.

> *"djatoka is a Java-based open source image server with an attractive basic feature set and extensibility under control of the community of implementers. Off-the-shelf, djatoka provides compression and region extraction of JPEG 2000 images, URI-addressability of regions, and support for a rich set of input/output image formats (e.g., BMP, GIF, JPG, PNG, PNM, TIF, JPEG 2000). djatoka also comes with a plug-in framework that allows transformations to be applied to regions and resolutions (e.g., watermarking)."* [70]

Il est conçu pour servir des images JPEG2000 (grâce à la librairie propriétaire kakadu [71]), mais est également compatible avec d'autres formats tels que TIFF.

Il propose deux viewers : le premier est basé sur IIPMooViewer, et le second sur OpenLayers. Ces deux viewers sont des versions modifiées afin de supporter le protocole d'échange avec le serveur djatoka (basé sur OpenURL [72]).
N.B. : La version IIPMooViewer 2.0 beta a récemment été modifiée afin de supporter le protocole djatoka. La compatibilité a donc été intégrée au projet original.

Pour aller plus loin :
- Fonctionnement de djatoka [73].
- Page de démonstration avec les viewers OpenLayer [74] et IIPMooViewer [75].

3.4.2.4 IIPImage

IIPImage est un logiciel serveur (*fast cgi module*) permettant de servir des *tiles* depuis des fichiers TIFF ou JPEG2000 (voir sections 3.5.4.1 et 3.5.4.3). Il se décrit comme :

> *"IIPImage is an Open Source light-weight streaming client-server system for the web-based viewing and zooming of ultra high-resolution images. It is designed to be bandwidth and memory efficient and usable over a slow internet connection."* [76]

70. http://sourceforge.net/apps/mediawiki/djatoka/index.php?title=Main_Page
71. http://sourceforge.net/apps/mediawiki/djatoka/index.php?title=Overview#JPEG_2000_as_the_Service_Format
72. http://sourceforge.net/apps/mediawiki/djatoka/index.php?title=Djatoka_OpenURL_Services
73. http://african.lanl.gov/aDORe/projects/djatoka/djatoka-jcdl09-poster.pdf
74. http://african.lanl.gov/adore-djatoka/openlayers/viewer.html?url=info:lanl-repo/ds/388ec2fd-049a-4195-910a-f0b2e73126aa
75. http://african.lanl.gov/adore-djatoka/viewer/viewer.html?url=info:lanl-repo/ds/388ec2fd-049a-4195-910a-f0b2e73126aa
76. http://iipimage.sourceforge.net/

Un script ou programme CGI (Common Gateway Interface) permet de déléguer la réponse d'un serveur web à un programme. Cette méthode n'est pas recommandée pour les sites/serveurs ayant un fort trafic, car chaque requête est un *fork* ; c'est-à-dire que chaque requête va être exécutée comme un processus séparé. Ceci n'est pas vraiment problématique si le temps d'exécution est court.

Par défaut, IIPImage ne permet que de servir des images au format TIFF pyramidal (PTIF). Il est possible d'ajouter très simplement le support de JPEG2000 en utilisant la version fournie par le site Old Maps Online [77]. Celle-ci utilise la libraire propriétaire kakadu [78] (gratuite à des fins non commerciales) pour le décodage des images au format JPEG2000 :

> *"The Kakadu software to convert TIFF to JPEG2000 is free, the part to decode it in the IIPImage server is not free but it is cheap."* [79]

> *"This choice, however, imposes a limit on the use of IIPImage when used with Kakadu : even though we are releasing our project as open-source software, the decoding functionality for the JPEG2000 format can be used only for non-commercial purposes due to the Kakadu library license. We are the Licensee of Kakadu library according the Kakadu Non-Commercial License and we distribute the Application to you (the Third party). The relevant paragraph from the license agreement is : "The Licensee shall have the right to Deployment of the KAKADU Software, provided that such Deployment does not result in any direct or indirect financial return to the Licensee or any other Third Party which further supplies or otherwise uses such Applications."* [80]

L'utilisation de l'un ou l'autre des formats (PTIF ou JPEG2000) avec IIPImage est transparente pour les utilisateurs, et n'a aucun impact au niveau du développement.

IIPImage peut servir les images via les protocoles **IIP** [81] et djatoka, ou générer **dynamiquement** des *tiles* aux formats Zoomify et Deep Zoom (voir section 3.5.4.4). N'importe quel viewer compatible avec l'un de ces protocoles peut donc être utilisé.

Une discussion sur le forum d'IIPImage sur ses avantages par rapport à des *tiles* statiques [82] implique qu'IIPImage est potentiellement plus rapide, car il met les *tiles* en cache sur le serveur. IIPImage est fourni avec plusieurs viewers [83] :

- IIPZoom – Flash Viewer
- IIPMooViewer – Ajax Javascript Viewer
- Jiip – Java Viewer
- Il est aussi possible d'utiliser n'importe quel logiciel tiers compatible avec les protocoles Zoomify, Deep Zoom, IIP ou djatoka.

N.B. : Ces viewers sont décrits à la section 3.4.3.

77. `http://help.oldmapsonline.org/jpeg2000`
78. Il existe plusieurs librairies pour le décodage de JPEG2000, mais leur analyse n'est pas utile pour ce travail. La plus aboutie semble être **kakadu** si l'on en croit le site Old Maps Online : `http://help.oldmapsonline.org/jpeg2000`.
79. `http://iipimage.sourceforge.net/2012/05/from-scan-to-delivery-special-collections-and-iipimage-at-utrecht-university-library/`
80. `http://help.oldmapsonline.org/jpeg2000/`
81. Internet Imaging Protocol :
    `http://www.i3a.org/technologies/digitalimaging/iip/`
    `http://iipimage.sourceforge.net/documentation/protocol/`
82. `http://sourceforge.net/projects/iipimage/forums/forum/299493/topic/3451410`
83. IIPImage demo : `http://iipimage.sourceforge.net/demo/`

Pour plus d'informations, *Object browsing using the Internet Imaging Protocol*[84] fait une synthèse des possibilités d'IIPImage Server.

Les "références" d'IIPImage sont également nombreuses[85]. Le serveur IIPImage est déjà utilisé par un grand nombre de musées, d'universités, etc.

3.4.3 Viewers (et possibilités d'annotations)

Cette section décrit les viewers compatibles avec les systèmes de *tiles* (qu'ils utilisent du JPEG2000, du PTIF ou des *tiles* statiques). Seuls les viewers fonctionnant dans un navigateur web sont traités ici, mais il est bon de noter qu'il en existe également sous forme d'application (Java, etc.). Les viewers permettant une personnalisation (*open source*) ou permettant l'utilisation de plusieurs types de *tiles* sont privilégiés. En outre, les possibilités d'annotation ou d'association de métadonnées sont également prises en considération.

La section *Have a look how your TIFF and JPEG2000 images can be presented on your website*[86] de l'excellent site *Old Maps Online* contient une liste de viewers compatibles avec IIPImage, avec pour chacun un lien vers une page de démonstration. Cette liste est très complète.

- IIPMooViewer
- Zoomify Viewer
- Seadragon Silverlight/AJAX
- OpenLayers
- OpenZoom
- PanoJS
- Seadragon application for Apple iPhone

3.4.3.1 Viewers non retenus

Pano JS Viewer (JavaScript + Zoomify)

Viewer simple qui offre très peu de fonctionnalités (malgré un zoom fluide) et dont la documentation est quasi inexistante.

OpenZoom (Flash + Zoomify)

OpenZoom SDK est un *toolkit* pour Adobe Flash Platform. Mentionné par souci de complétude, mais ignoré, car il n'est plus supporté.

> *"OpenZoom is no longer maintained nor supported."* [87]

Il était compatible avec les *tiles* au format Zoomify.

84. http://www9.org/w9cdrom/122/122.html
85. http://iipimage.sourceforge.net/links/
86. http://help.oldmapsonline.org/jpeg2000
87. http://www.openzoom.org/

Seadragon/Microsoft Zoom.it (JavaScript/Silverlight + Deep Zoom)

> *"Zoom.it (previously known as Seadragon.com) is a free* ***service*** *from Microsoft Live Labs that uses Deep Zoom image technology to tile and serve large images through an interactive zoomable viewer. [...] Zoom.it runs on Windows Azure and enhances the experience with Microsoft Silverlight, when available"* [88]

Sur la page github de *seadragon-ajax*

> *"Seadragon Ajax is a JavaScript library for viewing Deep Zoom Images. Among other things, it's what powers the image viewer on Zoom.it when your browser doesn't have Microsoft Silverlight installed. [...] Please note that Seadragon Ajax is no longer under active development or maintenance"* [89]

Le service Zoom.it est compatible avec le format Deep Zoom ou n'importe quelle image disponible sur internet et compatible avec Windows (PNG, JPEG, TIFF). Cette image doit cependant être accessible par une URL publique et visualisée au travers du service Zoom.it.
Le client seadragon-ajax n'est plus développé, puisque racheté par Microsoft.

Internet Archive BookReader (Javascript)

Le projet Open Library est un projet qui permet de visualiser des livres numérisés en ligne. Le viewer BookReader permet de passer en plein écran et de tourner les pages (avec un effet proche d'un livre réel). Les images ne sont par contre pas servies sous forme de *tiles*, ce qui rend leur affichage passablement lent.

> *" The Internet Archive BookReader [can run] on your own server [with] static images. [Others] have modified the IA BookReader to read image files from an image server, such as [Djatoka], instead of using static files on disk."* [90]

3.4.3.2 Zoomify (Flash + Zoomify)

Comme vu ci-dessus, Zoomify est une compagnie qui offre plusieurs solutions [91] pour l'affichage d'images en haute résolution sous forme de *tiles* au format JPEG ou PNG.

L'édition Zoomify Express [92] est gratuite et se compose d'un outil pour créer les *tiles* (Zoomify Converter), et d'un lecteur Flash. Cette édition convient pour un simple affichage d'images.

L'édition Zoomify Enterprise [93] est payante et offre des fonctionnalités supplémentaires :

- Annotation Viewer :

  > *"Zoomify Enterprise delivers Zoomify's powerful Annotation Viewer - now available in both HTML5 and Flash versions - providing sophisticated web-based image annotation including easily authored lists of Points Of Interest within an image and associated Notes and non-destructive in-image Label icons.* [...]*
  > *Polygon editing currently supported in HTML5 version of Annotation Viewer. Other editing features supported in Flash version of Annotation Viewer. See demos for full details."*

- Scriptable version of the Zoomify Converter.

88. http://www.microsoft.com/web/solutions/zoomit.aspx
89. https://github.com/aseemk/seadragon-ajax
90. http://raj.blog.archive.org/2011/03/17/how-to-serve-ia-style-books-from-your-own-cluster/
91. http://www.zoomify.com/compare.htm
92. http://www.zoomify.com/express.htm
93. http://www.zoomify.com/enterprise.htm

L'édition Zoomify Enterprise Developer [94] est plus chère et offre notamment en plus :

- *"Complete customization, enhancement, and integration control with full Flash source files for the Zoomify Annotation Viewer and Java source code for the Zoomify Servlet. (CS5 and CS4 users see note below***) [...]*
  *** ***Optional customization of the Zoomify Annotation Viewer requires Adobe Flash CS3 editor***. *Editing with Flash CS5 is not yet recommended [...]. "*
- *"Optional single-file storage (one zoomable '.PFF' file per image)"*
  N.B. : PFF = Pyramidal File Format (voir 3.5.4.2).

Concernant les possibilités d'annotations :

> ***"Is there any special server setup needed ?***
> *[...] For example, the Zoomify Hotspot Viewer can present data stored in XML (text) files, and the* ***Zoomify Annotation System can be used in an 'edit mode' that is designed to save edits with the help of a server-side 'posting' page.*** *"* [95]

Pour aller plus loin :

- Démo de *Full Annotation Editing* [96]
- Démo de *Side by Side Image Display with Linked Navigation* [97]
- FAQ de Zoomify, section Zoomify Annotation System (ZAS) [98]

3.4.3.3 IIPZoom (Flash)

IIPZoom est l'un des viewers proposés par IIPImage. Bien qu'il offre une fluidité très agréable au niveau du zoom, il dépend de Flash, qui doit être installé du côté client. Il est donc préférable d'utiliser IIPMooViewer qui est écrit en Javascript et qui ne nécessite rien d'autre qu'un navigateur web récent.

3.4.3.4 IIPMooViewer (Javascript + IIP/djatoka/Zoomify/Deep Zoom)

IIPMooViewer est un viewer écrit en JavaScript et développé pour le serveur IIPImage, avec lequel il fonctionne grâce au protocole IIP. La version stable (version 1.1) ne présente que peu d'intérêt. En revanche, la version 2.0 (actuellement en version beta) apporte des nouveautés très intéressantes ; on ne traitera donc pas de la version 1.1.

IIPMooViewer 2.0 beta est compatible avec les protocoles IIP et djatoka, ainsi que les *tiles* statiques au format Deep Zoom et Zoomify. Il n'est donc pas obligatoire de l'utiliser avec le serveur IIPImage. Outre les fonctions habituelles comme le zoom, le *panning*... la version 2.0 beta permet également :

- l'utilisation des quatre protocoles IIP, Zoomify, Deep Zoom, djatoka
- de pivoter les images (rotation)
- d'inclure des annotations (rectangles) sur les images

94. http://www.zoomify.com/enterprise.htm
95. http://www.zoomify.com/support.htm#a20081222_1435
96. http://www.zoomify.com/assets/dynamic/zasMedicalEditable.htm
97. http://digitisation.unimelb.edu.au/resources/publishing_examples/zoomify/side_by_side
98. http://www.zoomify.com/support.htm#q_ZEv3

Le serveur IIPImage a bonne presse, et IIPMooViewer est d'ores et déjà utilisé par un grand nombre de musées, d'universités, etc. [99]. La procédure d'installation est disponible en annexe (voir Annexe B.2.1).

Pour aller plus loin :

- Page de démonstration pour l'affichage d'annotations :

  > *"IIPImage javascript client demonstrates how annotations can be presented and edited, relative to zoomable images. This is an early example and the work is ongoing. A newer example, of annotation presentation can be seen here"* [100] [101]

- Blog d'IIPImage pour la version 2.0 beta d'IIPMooViewer [102]
- Page de démonstration avec des annotations [103]
- Page de démonstration avec deux viewers synchronisés [104]

3.4.3.5 OpenLayers (Javascript + Zoomify)

OpenLayers [105] est un viewer *open source* pour GIS (voir section 3.4.1). Il est compatible avec des systèmes tels qu'OpenStreetMap, mais aussi avec les *tiles* Zoomify [106]. Certains éléments d'annotations (plutôt axés GIS) existent déjà dans OpenLayers.

Pour aller plus loin :

- Exemple d'OpenLayers avec Zoomify [107]
- Page avec différents exemples [108]
- Exemple de compatibilité avec IIPImage [109]

N.B. : Une nouvelle version, OpenLayers 3 [110], est en cours de développement, mais n'en est qu'à ses balbutiements au moment de l'écriture de cet état de l'art [111].

3.4.4 Projets informatiques similaires

Outre les projets existants dans le domaine des *digital humanities* qui offrent en ligne la collection d'une bibliothèque, d'un auteur ou d'un artiste (voir 3.1), il existe quelques projets informatiques avec un champ d'application plus générique. Certains de ces projets sont payants, alors que d'autres mettent l'intégralité du code à disposition de la communauté de façon à ce qu'il puisse être réutilisé. Cette section décrit les quelques systèmes existants pouvant avoir un intérêt pour ce projet.

99. http://iipimage.sourceforge.net/links/
100. http://research.ng-london.org.uk/wiki/index.php/National_Gallery_IIPImage
101. http://cima.ng-london.org.uk/iip/annotation/
102. http://iipimage.sourceforge.net/2011/08/iipmooviewer-2-0-beta/
103. http://merovingio.c2rmf.cnrs.fr/iipimage/iipmooviewer-2.0/
104. http://merovingio.c2rmf.cnrs.fr/iipimage/iipmooviewer-2.0/synchro.html
105. http://docs.openlayers.org/library/introduction.html
106. http://blog.oldmapsonline.org/2008/06/openlayers-support-for-zoomify-first.html
http://iipimage.sourceforge.net/2008/06/zoomify-support/
http://blog.oldmapsonline.org/2008/06/imageserver-iipimage-now-supports.html
107. http://dev.openlayers.org/releases/OpenLayers-2.11/examples/zoomify.html
108. http://openlayers.org/dev/examples/
109. http://iipimage.mzk.cz/openlayers/
110. https://github.com/openlayers/ol3
111. http://openlayers.org/blog/2012/06/21/ol3-sprint/

3.4.4.1 BAMBI

BAMBI (Better Access to Manuscripts and Browsing of Images) [112] était un projet qui est passé en version commerciale et a été vendu sur un CD-ROM à la fin des années 1990.

> *"The project has produced a hypermedia system allowing historians, and more particularly codicologists and philologists, to read manuscripts, transcribe manuscripts, write annotations, and navigate between the words of the transcription and the matching piece of image in the numerized picture of the manuscript."* [113]

3.4.4.2 T-Pen

T-Pen (Transcription for Paleographical and Editorial Notation) : outil très complet qui permet la transcription de manuscrits. On peut l'utiliser en ligne gratuitement via le site officiel [114] ou installer sa propre version [115]. L'outil propose des aides comme la détection des lignes et la transcription ligne par ligne. Il est possible d'ajouter des annotations lors de la transcription (notes sur le texte transcrit, pas sur l'image). Les vidéos de démonstration montrent un outil très complet, mais doté d'une interface lourde et qui semble complexe à l'utilisation (beaucoup de menus et d'interfaces différentes) [116].

3.4.4.3 Diva.js

Diva.js (Document Image Viewer with AJAX) [117] est un projet permettant de visualiser, dans un navigateur web, plusieurs images en haute résolution comme un objet continu .

Les exigences de Diva.js sont proches des nôtres :

1. *Preserve Document Integrity*
2. *Allow side-by-side comparison of items*
3. *Provide multiple page resolutions*
4. *Optimize document loading*
5. *Present item and metadata simultaneously*

Ces 5 points font partie de l'analyse des besoins de cette thèse de Master, bien que le second n'entre pas dans le cahier des charges, et que dans le cas du cinquième il s'agisse d'annotations plutôt que des métadonnées.

L'affichage des pages se fait en colonne, les images les unes à la suite des autres (affichage en "rouleau"), grâce à une version modifiée du viewer IIPMooViewer, qui récupère les informations depuis une base de données. Les TIFF sont convertis en PTIF avec VIPS et les *tiles* sont servies avec le serveur IIPImage (protocole IIP).

Le projet en tant que tel ne correspond pas à nos besoins ; cependant l'état de l'art réalisé sur la problématique de l'affichage d'images en haute résolution, ainsi que la documentation sont tous les deux très complets [118].

112. http://elpub.scix.net/data/works/att/9817.content.pdf
113. http://journals.tdl.org/jodi/index.php/jodi/article/view/10/20
114. http://t-pen.org/TPEN/
115. https://github.com/jginther/T-PEN
116. http://www.youtube.com/user/tpentool
117. http://ddmal.music.mcgill.ca/diva
118. http://journal.code4lib.org/articles/5418

Pour aller plus loin :

- Page de démonstration [119].
- Projet sur github [120] et Wiki sur github [121]
- Setup and Installation (contains PTIF/VIPS/ImageMagick conversion) [122]

3.4.4.4 Lincolnnus/Image-Viewer

HTML5 Large Bio-Medical Image Viewer Project est un projet *Google Summer of Code 2012* pour l'affichage d'images biomédicales volumineuses [123]. Il est basé sur le serveur IIPImage et IIPMooViewer 2 (beta). Le projet se focalise sur le développement d'annotations et sur le *design* de la base de données, pour gérer les données des images. Les annotations se font avec un `canvas` HTML5 et il est possible de réaliser plusieurs types de formes (rectangle, ellipse, crayon et polyligne).

Ce projet correspond bien à nos besoins, car un bon travail a été réalisé pour la partie annotation.

3.4.5 Bilan intermédiaire

Dans cette section, la problématique de l'affichage d'images en haute résolution a été présentée. Toutes les solutions proposées reposent sur un système de *tiles*, qu'il s'agisse des GIS ou des serveurs d'images.

La partie "Serveur" permet de stocker et de servir les images à une application "Client". Celle-ci communique avec la partie "Serveur" afin de récupérer des portions d'images (*tiles*) et de les afficher de manière ordonnée. La partie annotation de l'image se fait du côté client pour l'interaction (sélection d'une zone) et l'envoi au serveur pour la sauvegarde (stockage de ladite annotation).

La présentation des systèmes d'affichage d'images a permis de mettre en avant le fait que ces trois parties (serveur, client et annotation) ne peuvent pas être considérées comme des éléments indépendants. En effet, les serveurs et les clients utilisent des protocoles bien précis pour communiquer, raison pour laquelle un viewer est généralement fourni ou recommandé par chaque solution serveur. Le choix de l'un dépend donc (souvent) directement de l'autre. Il en va de même pour l'annotation ; pour que cette dernière puisse se faire de pair avec le viewer, il faut que celui-ci permette les interactions de l'utilisateur et, idéalement, l'accès au code source. Ce projet requiert donc aussi une part importante de personnalisation du viewer, ou du moins la possibilité de pouvoir interagir avec lui, pour permettre les annotations.

3.4.5.1 Serveurs

GIS

Les GIS sont, parmi les systèmes existants, les plus approchants de ce que ce projet cherche à accomplir. En effet, pour la plupart ils offrent :

- d'afficher des images de grande taille sous forme de *tiles*

119. `http://ddmal.music.mcgill.ca/diva/demo/`
120. `https://github.com/DDMAL/diva.js`
121. `https://github.com/DDMAL/diva.js/wiki`
122. `https://github.com/DDMAL/diva.js/wiki/Installation#setting-up-the-backend`
123. `https://github.com/Lincolnnus/Image-Viewer`

- une interface de "navigation" : zoom, *panning* (déplacement avec la souris), etc.
- l'annotation, en liant une information à des points ou des zones (via des points, des polylignes, des polygones ou des points d'intérêts)

L'utilisation d'une métaphore d'interface [124] cartographique semble bien se prêter à ce projet, car il est aisé de considérer l'image d'un manuscrit comme une carte sur laquelle des annotations peuvent être liées à des zones (délimitées par des coordonnées "géographiques"). Cependant, après l'étude de la documentation et l'analyse de quelques solutions, il s'est avéré que les GIS sont souvent des logiciels complexes et difficiles à mettre en place sur un serveur privé.

De plus, ce sont des systèmes conçus pour stocker des informations cartographiques ou géographiques. L'utilisation d'un grand nombre de cartes individuelles (ici, les manuscrits) n'ayant aucun rapport entre elles n'est pas la fonctionnalité première d'un GIS, qui permet précisément de lier plusieurs jeux de données (comme pour la superposition de couches, afin de voir une carte vectorielle, puis l'image satellite correspondant aux mêmes coordonnées).

Bien que les GIS semblent être conceptuellement proches du but recherché, ils présentent des points négatifs permettant de les écarter :

- l'utilisation de coordonnées géographiques (type GPS) n'est pas très pratique. Une image possède déjà des coordonnées en deux dimensions (positions relatives à la hauteur et à la largeur).
- la mise en place d'un serveur privé est soit coûteuse, soit passablement laborieuse.
- les GIS sont généralement un groupe de logiciels (serveur web, base de données...) qui sont pensés pour des données géographiques. La modification, par exemple, de la base de données pour y intégrer une autre logique semble contre-productive.

Les questions qu'il faut alors se poser sont : l'effort conséquent pour mettre en œuvre un GIS, et son adaptation à la logique de ce projet sont-ils rentables par rapport aux résultats souhaités ? Dans quelle mesure est-il judicieux d'utiliser un système complètement axé sur la géographie, sachant que des systèmes gratuits (et beaucoup plus simples à mettre en place) existent pour l'affichage d'images en haute résolution ?

Il faut encore préciser que ce projet est développé par des informaticiens pour des linguistes qui ne sont pas censés avoir des connaissances informatiques avancées. La logique géographique inhérente au GIS devrait encore être "camouflée" pour éviter la confusion des utilisateurs finaux.

Vu la complexité des GIS et l'efficacité d'autres solutions pour servir les images (IIPImage par exemple), la métaphore d'interface cartographique semble alors inutilement laborieuse.

Serveurs d'images et systèmes basés sur des *tiles*

Deep Zoom ne présente pas d'avantage particulier pour nos besoins.

Zoomify offre des solutions intéressantes, mais toutes payantes lorsque l'on souhaite pouvoir les modifier. Au moment de l'écriture de cet état de l'art, les versions de Zoomify utilisant HTML5 sont encore en retard par rapport à leur équivalent basé sur Flash. Outre le prix de la licence Zoomify, la modification de l'application Flash nécessite encore l'achat d'un produit Adobe. Le système d'annotation se base sur un fichier XML par image, n'offrant pas des perspectives très ouvertes quant à l'indexation du contenu ou leur mise à jour (aucune centralisation de l'information).

Les serveurs IIPImage et djatoka permettent de servir des *tiles* depuis des fichiers PTIF et JPEG2000. Ils utilisent tous les deux la librairie kakadu pour le décodage du JPEG2000.

124. `http://fr.wikipedia.org/wiki/Métaphore_d'interface`

djatoka utilise également kakadu pour décoder les fichiers TIFF [125]. Cela implique que dans certains cas, des conversions intermédiaires ont lieu et celles-ci peuvent potentiellement poser des problèmes de mémoire [126].

Comme expliqué à la section 3.4.2.3, le serveur djatoka utilise son propre protocole qui n'est compatible qu'avec deux viewers, IIPMooViewer et OpenLayers. Le serveur IIPImage en revanche peut utiliser les protocoles IIP ou djatoka, et permet en plus de générer dynamiquement des *tiles* statiques (Zoomify, Deep Zoom), garantissant une compatibilité avec un grand nombre de viewers. Ceci est avantageux lors du développement, car avec un serveur IIPImage et une image, on peut tester la majorité des clients avec les différents types de *tiles*. On a vu également qu'IIPImage est très utilisé dans la pratique (voir section 3.1). Pour ces différentes raisons, le choix se tourne naturellement vers le serveur IIPImage.

3.4.5.2 Viewers et projets similaires

Le projet BAMBI est simplement trop vieux pour avoir un intérêt. Le projet Diva.js est une source intéressante pour la mise en place d'un serveur d'images et le choix du viewer, mais il y a trop de différences entre notre projet et Diva.js pour que celui-ci puisse être réutilisé. Le projet Lincolnnus/Image-Viewer (basé sur IIPMooViewer) semble quant à lui correspondre à ce que nous cherchons à réaliser ; du moins, il semble facilement adaptable à nos besoins. Malheureusement, ce projet a dû être exclu, car le code source disponible est moins exploitable qu'escompté (voir 6.7.3.1).

Comme aucun projet existant n'a pu être réutilisé, l'utilisation et la personnalisation d'un viewer restent la meilleure option. Le tableau de la figure 3.2 résume la compatibilité des viewers avec les protocoles et les formats d'image.

	Technologie		Format des *tiles*			
	Flash	Javascript	IIP	djatoka	Zoomify	DeepZoom
OpenZoom	x				x	
Zoomify Express	x				x	
Zoomify Enterprise	x	x			x	
Microsoft Zoom.it *		x				x
IIPZoom	x		x		x	x
IIPMooViewer		x	x	x	x	x
Pano JS Viewer		x			x	

* anciennement Seadragon

FIGURE 3.2 – *Comparaison des viewers*

On se rend rapidement compte qu'IIPMooViewer est le viewer qui offre le plus de flexibilité. Il peut être utilisé avec les serveurs IIPImage ou djatoka, mais aussi avec des *tiles* statiques comme Deep Zoom ou Zoomify, ce qui permet de ne pas se soucier du choix des autres technologies (serveur, format d'image), ni de risquer de bloquer l'évolution du système. IIPMooViewer est entièrement basé sur des technologies ouvertes et gratuites (HTML5/Javascript) et son code est *open source*. À l'instar du serveur IIPImage, on a vu qu'IIPMooViewer est très utilisé dans la pratique (voir section 3.4.4).

125. `http://sourceforge.net/apps/mediawiki/djatoka/index.php?title=Overview`
126. `http://sourceforge.net/mailarchive/forum.php?thread_name=38000f930910140757o73281c45rb2fb51f1484ab4ab@mail.gmail.com&forum_name=djatoka-general`

3.5 Images

3.5.1 Problématique

Comme on l'a vu dans la section 3.4, l'affichage d'images en haute résolution nécessite une application serveur pour servir des portions d'images à une application client, permettant ainsi une visualisation rapide sans systématiquement charger toute l'image. Ceci améliore grandement la fluidité lorsqu'on souhaite pouvoir agrandir/zoomer sur la zone d'intérêt en ayant une meilleure qualité d'image au fur et à mesure du zoom.

Pour ce faire, l'application serveur utilise des images sous une forme bien particulière. On parle alors de *(tiled) pyramidal images* (image pyramidale tuilée)[127], d'images pyramidales (pyramid images)[128][129], de *Tiled Multi-Resolution image*[130], de *multi-resolution network imaging*[131], de *zoomable high-resolution photos* ou images zoomables (*zoomable images*)[132], de *streamable images*, ... Autant de termes ou d'expressions qui définissent un même concept : **l'accès à des portions d'images à différentes résolutions**, mais représentant toujours une seule et même image.

3.5.2 *Tiles* (Tuiles)

L'affichage fluide d'images en haute résolution n'est pas une problématique récente et la solution la plus indiquée est d'utiliser un système de tuiles ou *tiles*.

> *"The individual images may be compressed or tiled. [Tiling] may yield more efficient delivery when most use involves zooming into image sections rather than viewing the whole."* [133]

L'image originale est découpée en grille (ou en mosaïque) où chaque élément est une portion de l'image appelée tuile ou *tile*. Un découpage similaire est ensuite répété à différentes résolutions. C'est-à-dire que plusieurs versions de la même image sont créées, et que chacune de ces versions possède une résolution plus petite que la précédente ; généralement, un facteur deux (2) est utilisé (comme illustré sur les figures 3.4 et 3.5). Chaque résolution représente un "niveau de zoom".

Pour rappel, la résolution d'une image est définie par le nombre de pixels en largeur et le nombre de pixels en hauteur (par ex. : 640x480). Plus la résolution est grande, plus l'image est précise ; plus elle est basse, plus la qualité de l'image diminue et plus l'on voit apparaître les pixels (voir figure 3.3).

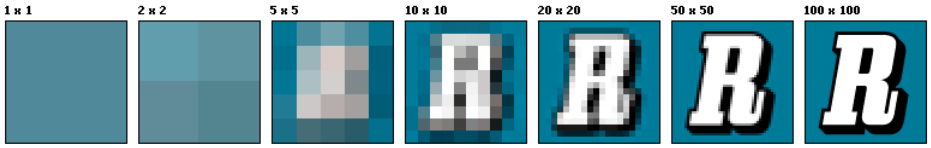


FIGURE 3.3 – *Une même image à plusieurs résolutions*[134]

127. http://iipimage.sourceforge.net/documentation/images/
128. http://en.wikipedia.org/wiki/Pyramid_(image_processing)
129. http://cs.haifa.ac.il/~nimrod/Compression/Image/I3prmd2005.pdf
130. http://www.digitalpreservation.gov/formats/fdd/fdd000237.shtml
131. http://www.zoomify.com/support.htm#a20081222_2108
132. http://www.microsoft.com/web/solutions/zoomit.aspx
133. http://www.digitalpreservation.gov/formats/fdd/fdd000237.shtml
134. image tirée de http://en.wikipedia.org/wiki/Image_resolution
135. image tirée de http://iipimage.sourceforge.net/documentation/images/

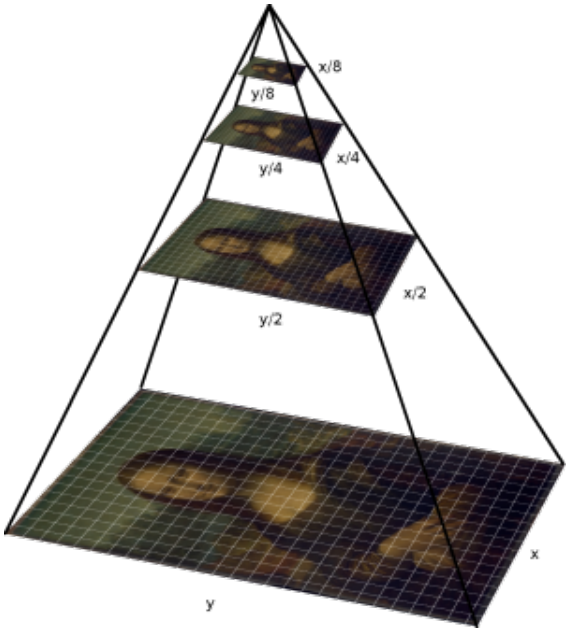


FIGURE 3.4 – *Image pyramidale* [135]

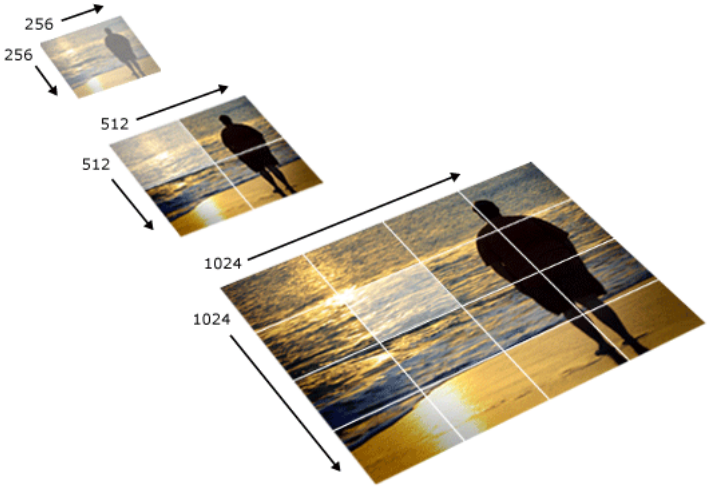


FIGURE 3.5 – *Découpage à plusieurs résolutions (1)* [136]

Le découpage en grille est donc appliqué à chacune des résolutions de l'image (voir figure 3.6) tout en gardant la même dimension de *tiles*, ceci quelle que soit la résolution de l'image.

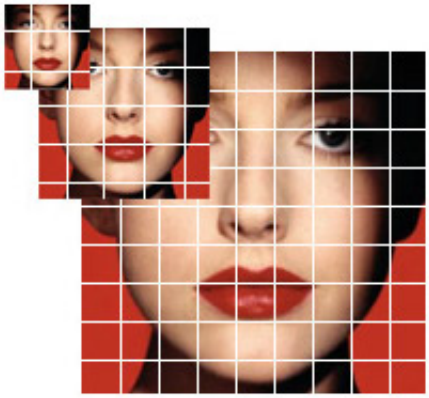


FIGURE 3.6 – *Découpage à plusieurs résolutions (2)* [137]

136. image tirée de `http://msdn.microsoft.com/en-us/library/cc645077%28v=vs.95%29.aspx`
137. image tirée de `http://www.zoomify.com/about.htm`

N.B. : Les *tiles* correspondant au bord droit et au bord inférieur de l'image peuvent être plus petites par le fait que la largeur/hauteur de l'image n'est que rarement un multiple de la largeur/hauteur d'une *tile* (voir figure 3.7). Ces *tiles* ont donc les dimensions :

$$largeur_{bord\_droit} = largeur_{image\_originale} \bmod largeur_{tile}$$

et

$$hauteur_{bord\_inférieur} = hauteur_{image\_originale} \bmod hauteur_{tile}$$

Le coin inférieur droit aura la dimension :

$$largeur_{bord\_droit} \times hauteur_{bord\_inférieur}$$

Ainsi, la version de l'image avec la plus haute résolution aura un nombre de *tiles* important, et l'image avec la plus petite résolution en aura très peu, voire une seule (voir figure 3.5).

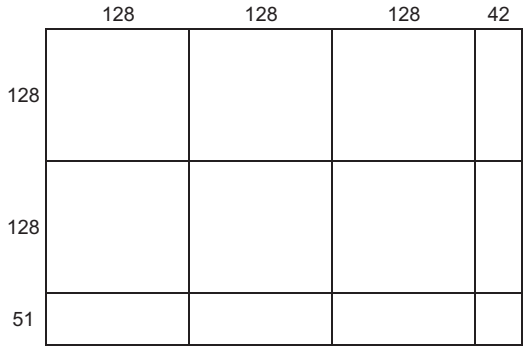


FIGURE 3.7 – *Découpage d'une image de 426x307 en tiles de 128x128 pixels*

Les différentes *tiles* peuvent être accessibles depuis :

- **un seul fichier**. Dans ce cas,
  - soit les *tiles* sont **stockées** dans un format *container* ; celui-ci contient l'image à différentes résolutions et découpées sous forme de tiles (Pyramidal TIFF - voir figure 3.4),
  - soit l'image originale (brute) est **encodée** dans un format permettant de directement **décoder** des *tiles* à différentes résolutions (JPEG2000 - voir figures 3.8 et 3.9).
- **un ensemble de fichiers**. Chaque *tile* existe sous la forme d'un fichier individuel et selon une structure (arborescence) bien précise (Zoomify, Deep Zoom).

À noter que dans tous les cas, les *tiles* sont générées *offline* (c'est-à-dire grâce à un traitement préalable à la mise en ligne). Ces trois cas sont détaillés dans la section 3.5.4.

Ce procédé de tuilage permet un affichage fluide de l'image, car plutôt que d'afficher l'image dans son entier, le viewer n'affiche simultanément qu'un certain nombre de *tiles*. Ce nombre dépend de la dimension du viewer et de la dimension des *tiles*, mais pas de la résolution (niveau de zoom). Ceci est illustré dans la Table 3.5 où l'on peut voir 4 niveaux de zoom (résolutions) de la même image affichés dans un viewer d'une dimension fixe.

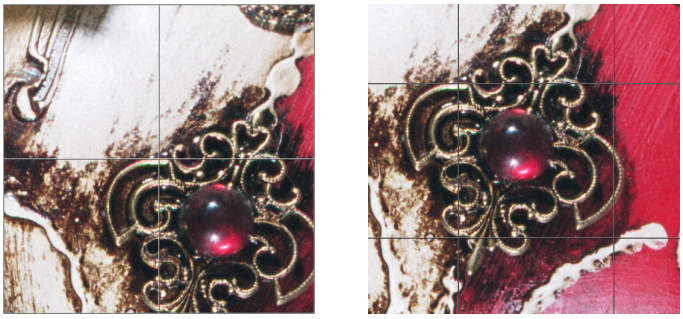
Dans cet exemple illustratif, le viewer affiche exactement 4 *tiles* dont la hauteur et la largeur s'ajustent parfaitement aux dimensions du viewer. C'est le cas idéal, car les *tiles* sont affichées dans leur intégralité. Rappelons que ceci est vrai, car toutes les *tiles* de l'image source ont les mêmes dimensions quel que soit le niveau de zoom (à l'exception des *tiles* des bords inférieur et droit, qui peuvent être plus petites). Dans un cas réel, le viewer devra souvent utiliser plus de *tiles*, car celles des bords ne seront que partiellement affichées. Ceci est le cas le plus courant. Il

Résolution 1 Résolution 2

Résolution 3 Résolution 4

TABLE 3.5 – *Exemple d'un viewer, dans un cas idéal, affichant 4 tiles à 4 résolutions*

est illustré dans les deux figures de la Table 3.6. La première figure (cas idéal) affiche exactement 4 *tiles* entières. La deuxième figure (cas courant) affiche seulement une *tile* en entier (celle du centre) et 8 partiellement.



Cas idéal Cas courant (avec décalage)

TABLE 3.6 – *Comparaison du cas idéal et d'un cas courant*

Pour un viewer d'une taille donnée, seul un nombre fini de *tiles* est chargé lors de la visualisation (quelle que soit la portion de l'image et le niveau de zoom). La quantité de données à télécharger pour visualiser une portion de l'image ne dépasse donc jamais un certain seuil (la somme de la taille des *tiles* affichées). Ce seuil est généralement bien en dessous de la taille de l'image originale. Le tableau 3.7 présente le volume de données à transférer (la somme des *tiles* en kilo octets) correspondant à chaque résolution des exemples des Tables 3.5 et 3.6.
N.B. : Précisons que la taille en KB de deux *tiles* de même dimension n'est pas nécessairement identique.

On constate qu'il devient rapidement intéressant d'utiliser des *tiles* (à partir de la résolution 2), et que plus la résolution augmente, plus le gain est significatif. Ainsi, cet exemple illustre bien le fait que, grâce aux *tiles*, il n'est pas plus "lourd" de visualiser l'image au zoom maximum

	Image totale (KB)	Tiles (nombre)	Portion d'image (KB)	Gain avec tiles (%)
Résolution 0 (thumbnail)	15.8	1	15.8	0.00
Résolution 1	55.2	4	55.2	0.00
Résolution 2	181	4	122	32.60
Résolution 3	555	4	112	79.82
Résolution 4	1372	4	74.9	94.54
Résolution 4 (avec décalage)	1372	9	164	88.05

TABLE 3.7 – Volume de données à transférer pour les exemples des Tables 3.5 et 3.6

plutôt qu'à un niveau de zoom inférieur. Si l'on se concentre sur l'exemple précis de la Table 3.5, le zoom maximum (résolution 4) est même significativement plus léger que les résolutions 2 et 3 (c'est un hasard, mais il se trouve que les *tiles* de la résolution 4 sont plus légères). Le volume de données téléchargé, dans l'exemple du cas idéal, reste en dessous de 122 KB ce qui est parfaitement raisonnable.

En comparant les volumes de données relatifs aux exemples de la Table 3.6, le cas courant est forcément moins bon que le cas idéal, mais donne malgré tout un excellent résultat avec un gain de 88.05%. Précisons, à toutes fins utiles, que la plus haute résolution de l'image dans cet exemple fait une taille de 1372 KB (1.34 MB), et que les photographies des manuscrits qui nous intéressent peuvent faire plusieurs dizaines de MB (jusqu'à 101 MB). Le nombre de résolutions peut également être plus important que 4.

3.5.3 Compression *lossless* or *lossy*

Les images peuvent être compressées ou non, qu'il s'agisse de *tiles* ou de l'image originale. De manière générale, le but de la compression d'un signal (un son, une image, une vidéo...) est de réduire la taille du fichier original, soit pour faciliter sa transmission/son utilisation, soit pour économiser de l'espace de stockage.

> *"Compression is used when storage space savings are significant"* [138]

On distingue deux types de compression :

la compression destructive (*lossy*) qui altère le signal original en supprimant l'information considérée comme non pertinente (ex : JPEG, PNG, MP3, DivX...). Le fichier compressé offre une information sémantiquement similaire, mais d'une qualité moindre et sa taille est plus petite (plus on perd d'information, plus la taille diminue). Ce procédé est **irréversible**, c'est-à-dire qu'on ne peut pas recréer le signal original depuis le fichier compressé.

la compression non destructive / compression sans perte (*lossless*) qui ne fait que compacter le signal, mais sans l'altérer. La taille du fichier est plus ou moins réduite selon l'algorithme utilisé, mais beaucoup moins qu'avec la compression destructive. Ce procédé est **réversible**, c'est-à-dire que le signal original peut être recréé depuis le fichier compressé.

138. `http://www.digitalpreservation.gov/formats/fdd/fdd000237.shtml`

Pour les images, l'un des formats de compression destructive le plus courant est le JPEG :

> *"The JPEG format was intended for use when the space a file uses has more value than the information it holds"* [139]

3.5.3.1 Quand compresser et quelle compression utiliser ?

Le cas qui nous intéresse ici est celui de la compression de l'image originale et des *tiles*.

Pour l'archivage de l'image originale, la compression *lossless* est l'unique option. Il est bien sûr également possible de garder le fichier original sans compression (fichier brut).

Pour les *tiles*, l'utilisation de la compression est par contre discutable. L'utilisateur doit accéder aux *tiles* depuis un navigateur web, et si celles-ci sont plus légères (compression destructive), l'information lui parviendra plus rapidement. Ceci est acceptable dans le cas de manuscrits numérisés si seule la lecture du texte manuscrit est importante. Cependant, s'il est déterminant de voir des détails pour une analyse plus fine (texture du papier, de l'encre...), alors la compression destructive devrait être évitée.

3.5.4 Formats d'image

Il existe différents moyens de créer des *tiles* et de stocker l'information. Cette section explique brièvement les formats auxquels il est fait référence dans ce travail.

3.5.4.1 TIFF

Le format TIFF (*Tagged Image File Format*) est un format d'image standard, mais très flexible.

> *"Il permet d'utiliser de nombreux types de compression, avec ou sans perte de données : brut, PackBits, LZW, CCITT Fax 3 et 4, JPEG. [...] Il permet le stockage d'image par bloc (on parle alors de TIFF tuilé), et aussi de multiples images par fichier, des images alternatives en basse résolution, des annotations sous forme de courbes et de texte, etc."* [140]

> *"For example, a TIFF file can be a container holding compressed (lossy) JPEG and (lossless) PackBits compressed images [...]."* [141]

Le format TIFF est le format d'archive de référence, car il permet de stocker des images brutes, mais aussi compressées sans perte (*lossless*). Il permet également de stocker plusieurs images (appelées pages) dans un même fichier (*multi-page file*), ce qui en fait un format *container*.

A noter qu'il existe une extension du format TIFF, appelée **GeoTIFF** [142], destinée aux systèmes d'information géographiques (GIS) (voir section 3.4.1). Ce format permet d'intégrer des informations géospatiales directement dans les métadonnées du fichier TIFF.

Les images sources utilisées dans ce projet sont des images TIFF non compressées.

139. http://vitaleartconservation.com/PDF/digital_image_file_formats_n_storage_v20a.pdf
140. http://fr.wikipedia.org/wiki/Tagged_Image_File_Format
141. http://en.wikipedia.org/wiki/Tagged_Image_File_Format
142. http://en.wikipedia.org/wiki/GeoTIFF

3.5.4.2 PTIF (Pyramid TIFF)

Le format PTIF (pour *Pyramid TIFF*, parfois aussi appelé *Tiled Multi-Resolution TIFF*, ou *Tiled Pyramidal TIFF*) est un cas particulier du fichier TIFF.
En effet, on a vu qu'un fichier TIFF pouvait contenir plusieurs images/pages. Un TIFF pyramidal stocke, au sein d'un même fichier, plusieurs versions d'une même image à des résolutions différentes. Chaque "sous-image" (page) est découpée en *tiles* (voir figure 3.4) :

> *"Tiled Multi-Resolution (or Tiled Pyramidal) TIFF is simply a tiled multi-page TIFF image, with each resolution stored as a separate layer within the TIFF."* [143]

> *"The term "Pyramid TIFF" is used to describe a TIFF file that wraps a sequence of bitmaps that each represent the same image at increasingly coarse spatial resolutions. The individual images may be compressed or tiled. Compression is used when storage space savings are significant ; tiling may yield more efficient delivery when most use involves zooming into image sections rather than viewing the whole."* [144]

Ce format est pratique, car il permet de stocker dans un seul fichier, plusieurs pages :
- (optionnel) l'image originale (brute ou compressée *lossless*).
- la "pyramide" des images de résolutions moindres (comme illustré à la figure 3.4). Chaque étage de la pyramide est une page du fichier TIFF ; elle peut être compressée ou non.

Sur la base d'une image TIFF, on peut créer une image pyramidale (PTIF) (voir Annexe B.1.1).

La compagnie Zoomify offre également un système *container* équivalant au PTIF : Zoomify's Pyramidal File Format (PFF).
La lecture de ce type de fichiers nécessite un *servlet* Java sur le serveur web [145].

3.5.4.3 JPEG2000

Le JPEG2000 est un format d'image créé par le même groupe que son prédécesseur le JPEG [146]. Grâce à un complexe algorithme de compression [147] (permettant la compression destructive ou non destructive), le JPEG2000 offre une plus grande flexibilité que le JPEG classique.

> *"The codestream obtained after compression of an image with JPEG 2000 is scalable in nature, meaning that it can be decoded in a number of ways ; for instance, by truncating the codestream at any point, one may obtain a representation of the image at a lower resolution, or signal-to-noise ratio – see scalable compression. By ordering the codestream in various ways, applications can achieve significant performance increases. However, as a consequence of this flexibility, JPEG 2000 requires encoders/decoders that are complex and computationally demanding."* [148]

143. `http://www.digitalpreservation.gov/formats/fdd/fdd000237.shtml`
144. `http://www.digitalpreservation.gov/formats/fdd/fdd000237.shtml`
145. `http://www.zoomify.com/support.htm#a20081222_2108`
146. Les deux formats tirent leur nom du groupe *Joint Photographic Experts Group* qui a créé ces normes.
147. `http://en.wikipedia.org/wiki/JPEG_2000#Technical_discussion`
148. `http://en.wikipedia.org/wiki/JPEG_2000`

Le JPEG2000 **encode** l'information de l'image originale et l'organise de manière flexible et intelligente, permettant une compression efficace (diminution de 30%-70% en *lossless*, et jusqu'à 300 fois plus avec la compression destructive) [149].

Cette flexibilité résulte notamment de l'utilisation, dans l'algorithme de compression, d'une décomposition en ondelettes (*discrete wavelet transform*) [150] là où le JPEG utilise une transformée en cosinus discrète (*discrete cosine transformation*) [151]. La figure 3.8 montre la transformation (encodage) d'une image en JPEG2000. Lors du décodage, plusieurs "niveaux" d'information peuvent être obtenus grâce à la décomposition en ondelettes, comme illustré à la figure 3.9 : on peut directement **décoder** une image entière ou une portion d'image (soit avec des *tiles*, soit avec des *precincts*), et ceci à différentes résolutions.

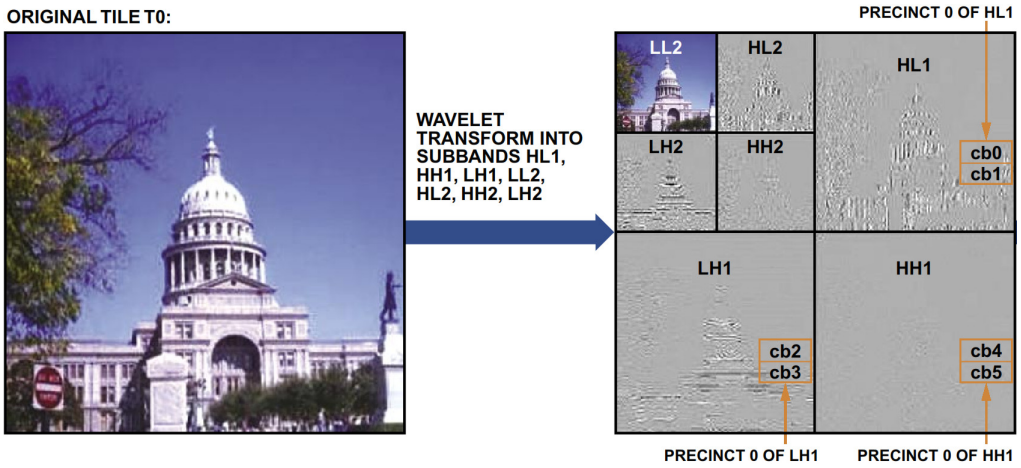


FIGURE 3.8 – *JPEG2000 Encode - Image over wavelet transform into subbands and resolutions* [152]

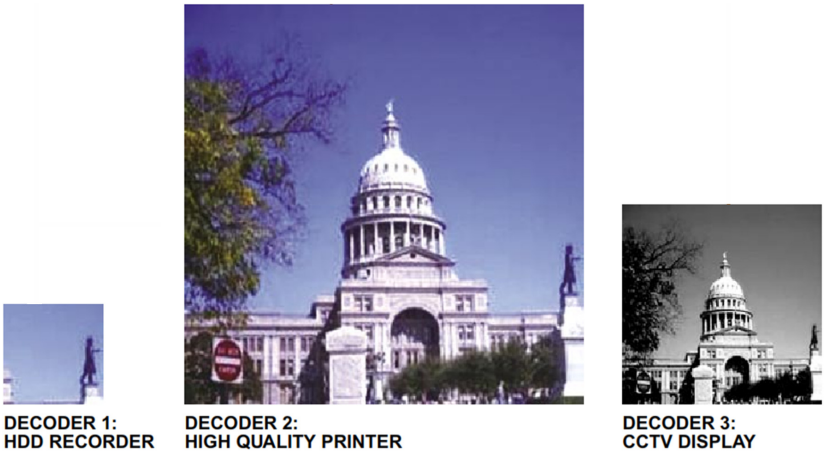


FIGURE 3.9 – *JPEG2000 decode - one JPEG 2000 stream is received by several decoders* [153]

Les *precincts* sont comparables à des *tiles*, mais dans le domaine des ondelettes :

> *"Each resolution of a tile-component is partitioned into precincts. Precincts behave much like tiles, but in the wavelet domain. The precinct size can be chosen independently by resolution ; however, each side of a precinct must be a power of 2 in size. [Figure 3.10] shows a precinct partition for a single resolution.*

149. `http://vitaleartconservation.com/PDF/digital_image_file_formats_n_storage_v20a.pdf`
150. `http://en.wikipedia.org/wiki/Wavelet_transform`
151. `http://fr.wikipedia.org/wiki/JPEG#Transformée_DCT`
152. image tirée de `http://www.analog.com/library/analogDialogue/archives/38-09/jpeg2000.pdf`
153. image tirée de `http://www.analog.com/library/analogDialogue/archives/38-09/jpeg2000.pdf`

Precincts are another ingredients to low-memory implementations in the absence of tiles. Compressed data from a precinct are grouped together to form a packet. Before a packet header can be created, all compressed data for a corresponding precinct must be available. Thus, only the compressed data for a precinct must be buffered, rather than the data of an entire tile (or image in the absence of tiles).” [154]

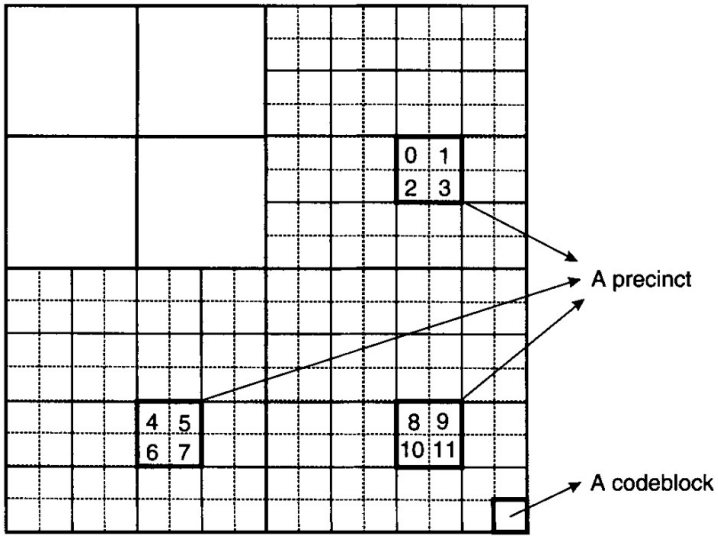


FIGURE 3.10 – *Partitioning of wavelet subbands* [155]

Le schéma de la figure 3.10 est illustré avec un exemple concret à la figure 3.11.

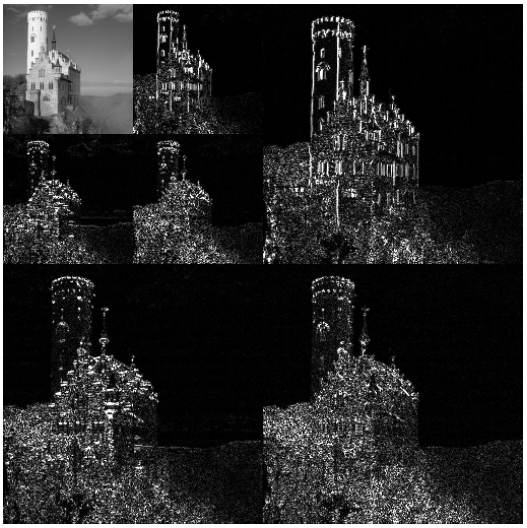


FIGURE 3.11 – *Exemple de precinct (1)* [156]

“A precinct represents a spatial region within a given "resolution level" of a given image component. Each resolution level builds upon the previous resolution level to offer the additional details required for increased reconstructed image resolution.” [157]

154. `http://books.google.ch/books?id=LjQiGwyabVwC&pg=PA358&lpg=PA358`
155. image tirée de `http://iipimage.sourceforge.net/documentation/images/`
156. image tirée de `http://en.wikipedia.org/wiki/JPEG_2000`
157. `http://www.kakadusoftware.com/documents/jpik.pdf`

L'algorithme de compression du JPEG2000 ainsi que la théorie l'accompagnant sortent largement du cadre de ce travail et ne seront pas détaillés plus en avant. Il est surtout important de retenir que le **décodage** d'un JPEG2000 permet d'accéder à l'image entière ou des parties de l'image, et ceci à plusieurs résolutions.

Le décodage implique qu'un traitement doit être fait lors de la lecture de l'image [158]. Lorsque le décodage est fait par un serveur web, cela peut provoquer des ralentissements si plusieurs requêtes interviennent simultanément, mais des solutions existent pour prévenir ce problème [159].

À noter que le décodage d'un JPEG2000 produit une autre image, mais qui sera également dans le format JPEG2000. Celle-ci sera, par exemple, de moindre qualité ou représentera une portion de l'image originale. Comme peu de navigateurs web peuvent afficher du JPEG2000, il est généralement nécessaire de convertir cette image de moindre qualité dans un format largement compatible (JPEG ou PNG) avant que le serveur ne l'envoie au navigateur [160].

Bien que la notion de *tiles* fasse partie des spécifications de JPEG2000, il semble qu'il soit préférable d'utiliser les *precincts* :

> *"Having tiles in JPEG2000 means about the same as in TIFF. The original image is [divided] in blocks which are compressed totally individually. This is easily giving you nasty visible boundaries between the tiles if you have much compression. [...] Unlike in TIFF, tiles are not needed in order to get a fast spatial random access to JPEG2000 image. A suitable progression order and another mechanism called "precincts" can handle that without any tiles and annoying tile boundaries."* [161]

Résumé clair des fonctionnalités principales de JPEG2000 :

> *"The JPEG 2000 image format has attracted considerable attention among others due to its open specification, and its rich feature set defined in a multi-part ISO standard. Features of interest in the context of this article are :*
> *- Multiple Resolutions : When compressing a master image, JPEG 2000 uses the multi-resolution attributes of the discrete wavelet transform (DWT) to store multiple DWT levels. In djatoka, the number of DWT levels is determined by the number of times an image can be halved from its maximum resolutions longest side to 92 pixels or less.*
> *- Region Extraction : The structure of JPEG 2000 files supports region decoding such that only the image data needed to render a sub-region is decompressed for display.*
> *- Compression : Support for lossless and lossy wavelet-based compression.*
> *- Self-containdness : Support for embedding XML metadata within the image bitstream, which may be used for licensing information, provenance description, descriptive metadata, etc.*
> *- Progressive Transmission : This allows a viewer to display a lower quality version of the image as soon as a small part of the entire image file has been received ; the image quality then improves progressively as the downloading proceeds. JPEG 2000 supports progressive transmission in two dimensions : by pixel accuracy and by image resolution."* [162]

Il est possible de convertir une image TIFF, sans perte, en JPEG2000 (voir Annexe B.1.2).

158. `http://en.wikipedia.org/wiki/JPEG_2000`
`http://faculty.gvsu.edu/aboufade/web/wavelets/student_work/EF/how-works.html`
159. voir la section "Tile Image Server" de la documentation de Diva.js :
`http://journal.code4lib.org/articles/5418`
160. Section 4.3 support : `http://wellcomelibrary.org/assets/wtx056572.pdf`
161. `http://tech.groups.yahoo.com/group/kakadu_jpeg2000/message/3615`
162. `http://sourceforge.net/apps/mediawiki/djatoka/index.php?title=Overview`

Pour plus d'informations sur le JPEG200 :

- JPEG 2000 Image Compression [163]
- The JPEG2000 Image Compression Standard [164]
- Lossless Compression Handbook [165]

3.5.4.4 *Tiles* statiques (arborescence)

Une méthode également très courante consiste à générer une arborescence contenant un fichier descripteur (XML) et un ensemble de *tiles*, où chaque *tile* est sauvée comme un fichier image. Le nom du fichier image (et éventuellement le chemin dans l'arborescence) représente la position (résolution, colonne et ligne) de la *tile* dans une grille correspondant à la résolution (voir figure 3.12).

Cette approche de *tiles* statiques est utilisée par Zoomify [166] et Microsoft Deep Zoom [167]. Notons encore, par souci de complétude, que Google utilise un système similaire pour *Google Maps* et *Google Earth*.

Zoomify

Voici un exemple d'arborescence compatible avec Zoomify. Un fichier XML contient les informations sur les dimensions de l'image, le nombre de *tiles* au total, la résolution d'une *tile*, etc.

ImageProperties.xml Ce fichier contient :

```
<IMAGE_PROPERTIES WIDTH="2080" HEIGHT="3120" NUMTILES="169" NUMIMAGES="1"
VERSION="1.8" TILESIZE="256" />
```

Dans le même répertoire contenant "ImageProperties.xml" se trouve un sous-répertoire (*TileGroup0*) contenant les *tiles* au format JPEG (ici, 169 fichiers).

ImageProperties.xml

TileGroup0\0-0-0.jpg Résolution 0 = Thumbnail

TileGroup0\1-0-0.jpg Résolution 1, colonne 0, ligne 0

TileGroup0\1-0-1.jpg Résolution 1, colonne 0, ligne 1

TileGroup0\1-1-0.jpg Résolution 1, colonne 1, ligne 0

TileGroup0\1-1-1.jpg Résolution 1, colonne 1, ligne 1

TileGroup0\... ...

TileGroup0\2-1-2.jpg Résolution 2, colonne 1, ligne 2

TileGroup0\... ...

TileGroup0\4-8-12.jpg Résolution 4, colonne 8, ligne 12

163. http://www.analog.com/library/analogDialogue/archives/38-09/jpeg2000.pdf
164. http://webdocs.cs.ualberta.ca/~anup/Courses/604/NOTES/slide_jpeg2000.pdf
165. http://books.google.ch/books?id=LjQiGwyabVwC&pg=PA358&lpg=PA358
166. http://www.zoomify.com/support.htm#a20081218_2335
167. http://msdn.microsoft.com/en-us/library/cc645077%28v=vs.95%29.aspx

(a) *Résolution 2* (b) *Résolution 2 avec numérotation des tiles*

FIGURE 3.12 – *Zoomify tiles*[168]

Deep Zoom Image

Deep Zoom est une technologie développée par Seadragon, puis rachetée par Microsoft, qui permet la visualisation d'images en haute résolution.

Le concept de génération des tiles est proche de Zoomify : une description en XML, avec une extension *.xml ou *.dzi (= Deep Zoom Image), ainsi qu'une arborescence où chaque niveau/résolution est un sous-répertoire et chaque *tile* est un fichier séparé au format JPEG ou PNG[169] :

monImage.xml
monImage_files\0\0_0.jpg Résolution 0 = Thumbnail
monImage_files\1\0_0.jpg Résolution 1, colonne 0, ligne 0
monImage_files\1\0_1.jpg Résolution 1, colonne 0, ligne 1
monImage_files\1\0_2.jpg Résolution 1, colonne 0, ligne 2
monImage_files\... ...

Ce format propose des choses intéressantes, comme la possibilité de créer des *sparce image*[170] (où certaines zones peuvent avoir une résolution plus élevée que le reste de l'image, permettant de zoomer encore davantage sur une zone précise), ou la notion de collection[171] (groupe d'images).

3.5.4.5 Test comparatif

Voici quelques comparaisons récoltées sur internet :

- Le test *JPEG2000 vs JPEG (vs TIFF)*[172] compare la qualité de plusieurs images générées avec différentes options de compression, ainsi que la présence d'éventuels artefacts ; le test se conclut par une comparaison des tailles de fichiers :

  > *" JPEG2000 vs TIFF Thanks to its 48bit color mode JPEG2000 is also serious competitor for the ancient but still wide-spread TIFF format.*
  > *I did a short test with a large image file (5443x3636 pixel, 16bit color depth) :*
  > *TIFF 116 MB*
  > *TIFF with LZW compression 150 MB (which proves that LZW is ineffective for image compression)*
  > *JPEG2000 - lossless 61 MB*
  > *JPEG2000 - 100% quality (lossy mode) 21 MB "*

168. images tirées de `http://www.zoomify.com/`
169. `http://msdn.microsoft.com/en-us/library/cc645050(v=vs.95).aspx#creating_a_deep_zoom_image`
`http://www.gasi.ch/blog/inside-deep-zoom-2/`
`http://blogs.msdn.com/b/jaimer/archive/2008/03/31/a-deepzoom-primer-explained-and-coded.aspx`
170. `http://msdn.microsoft.com/en-us/library/cc645077(v=vs.95).aspx#Sparse_Images`
171. `http://msdn.microsoft.com/en-us/library/cc645077(v=vs.95).aspx#Collections`
172. `http://www.photozone.de/jpeg2000-vs-jpeg-vs-tiff`

- Le Wiki de Diva.js contient une comparaison de différents paramètres de compression en JPEG2000, avec le programme kdu_compress (Kakadu), et des portions d'images à comparer [173].
- Le document *British Library : JPEG 2000 profile* [174] recommande des paramètres de compression.
- L'Université d'Utrecht a fait des estimations de temps de conversion entre PTIF et JPEG2000, déclarant JPEG2000 comme vainqueur :

  > *“* ***Images***
  > *Next step was the automation of conversion of existing and daily scans to tiled pyramidal TIFFs. We already used ImageMagick so we started a test batch of converting TIFFs each approx. 100G in size. [...] and it would probably take a year or more to convert the whole set. Alas, switching to VIPS made no significant difference. [We] decided to go for another option : convert to JPEG2000 based on Kakadu software. [...] The performance improved spectacularly [...]. The Kakadu software to convert TIFF to JPEG2000 is free [and] within a few weeks we converted the whole set of 1,361,067 TIFFs to JPEG2000 [...].* ” [175]

- Comparaison de JPEG2000 vs. TIFF vs. JPEG, ainsi qu'une discussion sur les problématiques de stockage [176].

3.5.5 Bilan : comparaison TIFF - JPEG2000 - *tiles* statiques

Ce travail ne consiste pas à étudier et comparer les différents formats d'images dans le détail. Cependant, les manuscrits étant au format TIFF non compressé, un traitement est de toute façon indispensable pour convertir le TIFF original sous une forme permettant d'accéder à des *tiles* : ce traitement préalable (*offline*) sert à découper les *tiles* à différentes résolutions.

Les *tiles* statiques ont l'avantage d'utiliser un mécanisme simple et qui ne demande aucune configuration particulière sur le serveur. Leur principal désavantage réside dans le fait qu'il est moins aisé d'administrer des arborescences plutôt qu'un ensemble de fichiers uniques. De plus les *tiles* sont toujours dans un format compressé avec perte (*lossy*) [177]. Ces formats ne peuvent donc pas servir comme format d'archivage.

Le débat est encore ouvert entre TIFF/PTIF et JPEG2000 [178] : le format TIFF existe depuis longtemps et on le sait fiable pour l'archivage d'images non compressées ; la transformation en PTIF permet d'accéder à des *tiles*. Le JPEG2000 étant avant tout un format de compression, l'avantage principal par rapport au PTIF est un gain en termes d'espace de stockage : il offre un système de compression (notamment non destructive (*lossless*)) très efficace et permet de réduire le poids des images (voir section 3.5.4.5). De plus, JPEG2000 utilise nativement des *tiles* (ou des *precincts*) et permet aussi d'accéder à une image de basse résolution depuis une image maître (*master image*) [179] contenue dans un fichier unique.

173. https://github.com/DDMAL/diva.js/wiki/JPEG-2000-encoding
174. http://www.digitizationguidelines.gov/still-image/documents/Martin.pdf
175. http://iipimage.sourceforge.net/2012/05/from-scan-to-delivery-special-collections-and-iipimage-at-utrecht-university-library/
176. http://vitaleartconservation.com/PDF/digital_image_file_formats_n_storage_v20a.pdf
177. http://www.zoomify.com/support.htm#a20081222_1827
178. http://www.dpconline.org/advice/faqs/588-faq-tiff-or-jpeg2000 http://www.dlib.org/dlib/november09/kulovits/11kulovits.html
179. http://faculty.gvsu.edu/aboufade/web/wavelets/student_work/EF/how-works.html

La conversion d'une image TIFF en JPEG2000 est très rapide par rapport à une conversion de TIFF à TIFF pyramidal (coûteuse en temps), et permet également d'économiser de l'espace de stockage.

Malgré tous ses avantages, le JPEG2000 doit être décodé et donc il nécessite un effort computationnel pour le décodage (voir section 3.5.4.3) ***a priori*** plus important que le PTIF. Cependant, les fichiers PTIF et JPEG2000 nécessitent tous les deux un logiciel au niveau du serveur web afin de servir des *tiles* à une application client ou au navigateur. Si le serveur est vraiment très peu puissant, alors il est préférable d'utiliser des images au format PTIF. La nécessité de devoir décoder le JPEG2000 est l'une des raisons principales pour laquelle le JPEG2000 est encore un peu boudé à des fins d'archivage, et que TIFF reste plus répandu (bien que cette tendance commence à changer).

Selon les estimations fournies par les utilisateurs du projet sur le nombre total de feuillets, il pourrait y avoir, à terme, environ entre 250'000 et 300'000 images. Il sera donc plus aisé de gérer un corpus d'images uniques (PTIF, JPEG2000), où chaque fichier sert à la fois d'archive et de source de consultation, plutôt que des arborescences de *tiles* statiques (Zoomify, Deep Zoom).

Quelques rapides tests et les références de la section 3.5.4.5 permettent de constater que PTIF et JPEG2000 offrent une qualité suffisante pour nos besoins. Les critères s'arrêtent donc sur le temps de conversion, le gain d'espace disque si la compression est utilisée et la facilité de gestion (un fichier *container* vs. une arborescence).

En conclusion, le format JPEG2000 est le plus adapté à ce projet, car il offre un temps de conversion court, une excellente qualité, une taille de fichier plus petite que la moyenne et l'accès aux *tiles* via un fichier unique. Les différentes options pour la partie serveur sont décrites à la section 3.4.2.

Chapitre 4

Modèle de données

L'analyse des besoins des utilisateurs (chapitre 2) a permis de déterminer comment structurer l'information nécessaire à ce projet. En effet, cette structure dépend notamment du type d'information que l'on souhaite extraire avec des requêtes de recherche. Le modèle doit être suffisamment expressif pour pouvoir extraire, par exemple, les annotations liées à un manuscrit, faire de la recherche dans certains textes, etc.

Cette section présente la structure du modèle et l'illustre avec un diagramme UML et un schéma RDFS. Les concepts spécifiques du domaine sont détaillés et définis ici. Pour ôter toute ambiguïté, plusieurs notions relatives à la formalisation des données ont été définies en accord avec les utilisateurs du projet. On les retrouve dans le RDFS et l'UML.

4.1 Concepts du domaine

Dans cette section nous présentons les principaux concepts du domaine de l'analyse scientifique des manuscrits.

4.1.1 Annotations

Selon le Trésor de la langue française :

> *"[Annoter :] Accompagner un texte de remarques diverses généralement manuscrites.*
> *En partic. Accompagner un texte publié de notes et remarques critiques pour en commenter ou interpréter les passages douteux ou obscurs."* [1]

Dans ce travail, nous ne traitons que de l'**annotation graphique** sur les images des manuscrits. On la définit comme étant un texte lié à une zone délimitée graphiquement sur une image. Nous considérons que les utilisateurs vont employer les annotations pour :

- des **notes** (post-it / note dans la marge),
- des **commentaires / précisions**,
- des **liens** (hyperliens URL) vers des données de fond (page web, article, PDF...).

1. `http://atilf.atilf.fr/`

"[A manuscript] makes references to people, places, and events that are unfamiliar to you or to other people who might read your transcription. Part of the editing of an original manuscript can involve doing the background research and providing explanations of those references. Particularly in a family manuscript, identifications of people and family events is especially important. This can be a challenging part of the editing, because writers of diaries, journals, and letters assume knowledge on their own parts or on their correspondents' parts that does not require explanation–a mere allusion will suffice." [2]

Le texte lié à la zone est un champ libre permettant de saisir du texte formaté en HTML. Il a été convenu que l'HTML correspondait aux besoins des utilisateurs, car il permet d'afficher du texte formaté (titre, gras, souligné, barré...), et des liens hypertextes (URL) vers des pages web ou d'autres documents numériques (PDF, diaporama...). Toutefois, d'autres formats auraient pu être utilisés (Wiki markup, RTF...).

En ce qui concerne la délimitation graphique d'une zone d'annotation sur l'image, toutes formes géométriques en une ou deux dimensions pourraient servir de délimiteur. On peut s'inspirer des systèmes GIS (voir section 3.4.1) qui utilisent des *Hot spot*, des *Points of interest (POI)* et des zones géométriques (rectangle, polygone, points, polylignes...).

Vraisemblablement, les rectangles et les polylignes sont les types de formes les mieux adaptées à l'annotation de manuscrit. Dans ce travail, on se limite aux zones rectangulaires dans un premier temps afin de permettre une bonne compatibilité avec le viewer IIPMooviewer (voir 6.7.3).

4.1.2 Transcriptions

Trésor de la langue française :

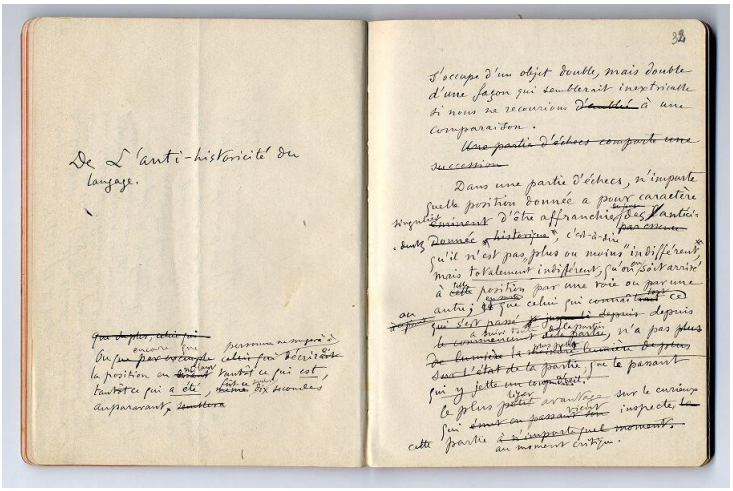
" A. [Corresp. à transcrire A] Reproduction exacte, par l'écriture, de ce qui a déjà été écrit ; résultat de cette action.
B. [Corresp. à transcrire B] Reproduction exacte par écrit à l'aide d'autres signes, d'un système de notation différent, d'un autre code. 1. LING. Représentation d'unités phoniques ou graphiques au moyen de signes, d'un alphabet, d'une écriture différents. " [3]

Le contenu d'une transcription, comme pour les annotations, sera uniquement du texte formaté en HTML. Outre la facilité de saisie pour l'utilisateur, la mise en forme du langage HTML permet de faire une transcription visuellement proche du manuscrit (souligner, barrer du texte...). Pour les cas impossibles à représenter visuellement (insertion, etc.), les codes de transcription habituels des linguistes seront utilisés. Bien qu'envisagé, l'encodage au format TEI n'est pas retenu dans ce travail (voir section 3.2.3).

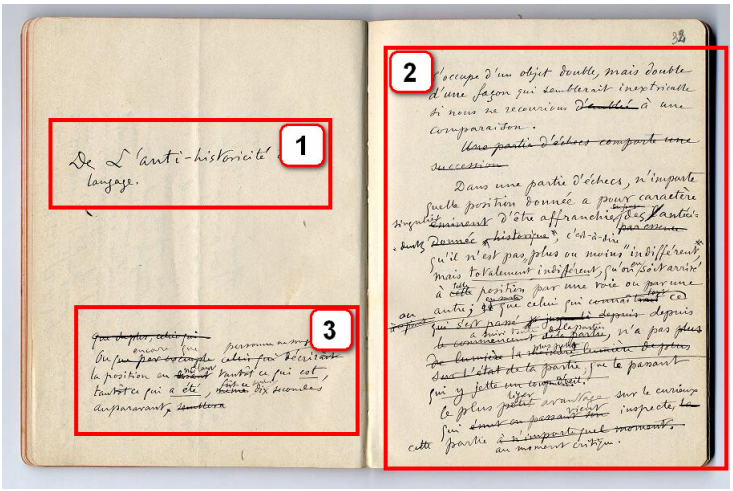
Cependant, il faut préciser qu'une transcription feuillet par feuillet n'a pas forcément de sens dans le cas des manuscrits de Ferdinand de Saussure. En effet, celui-ci écrivait sur différents types de supports, n'hésitant pas à les pivoter, les retourner jusqu'à ce qu'ils soient complétement remplis. Les ratures sont aussi monnaie courante dans ses écrits. Par exemple, lorsqu'il utilisait des cahiers, il gardait parfois vierge la page de gauche, afin de pouvoir y apposer des idées au cours de la rédaction ; elle lui servait aussi à insérer des ajouts dans le texte principal, situé sur la page de droite. Le tableau 4.1 présente deux versions de deux pages d'un cahier : une version "brute" et une version avec l'ordre de lecture "correct". Il existe des cas plus complexes, comme

2. `http://www.chsbs.cmich.edu/Robert_Root/Guide/Annot.htm`
3. `http://atilf.atilf.fr/`

un faire-part sur lequel Ferdinand de Saussure a écrit tout autour, ou un papier d'emballage de grande dimension qu'il a utilisé pour écrire et structurer ses idées et avoir, littéralement, une vue d'ensemble.



Version brute



Version avec ordre de lecture

TABLE 4.1 – *Feuillet* `MS.FR. 3951 / 10 f. 32-33`

Ainsi, afin de définir un fil de lecture plus cohérent, la transcription est définie par une **séquence d'éléments de transcription**. À l'instar d'une annotation, chaque élément de transcription est lié à une zone. Il est ainsi possible de créer :

- des transcriptions dites **physiques** ou **typographiques**, ne tenant pas compte de l'ordre "correct" de lecture. Elles correspondent littéralement à la transcription du contenu d'un feuillet (= séquence de tous les éléments de transcription du feuillet). Par ex. : ordre 1, 3, 2 dans le tableau 4.1.
- des transcriptions dites **logiques**, correspondant à l'ordre dans lequel le texte est censé être lu pour permettre une lecture cohérente. Par ex. : ordre 1, 2, 3 dans le tableau 4.1.

Plusieurs transcriptions peuvent exister si elles utilisent des ordres distincts. En effet, il peut y avoir différentes interprétations quant à l'ordre de lecture, car il dépend d'une appréciation qui reste après tout subjective [4].
N.B. : Comme illustré à la figure 4.1, une transcription **logique** n'exclut pas que des éléments de transcription soient répartis sur plusieurs feuillets.

4. Notons que la question de l'ordre est un sujet de désaccord fréquent chez les saussuriens.

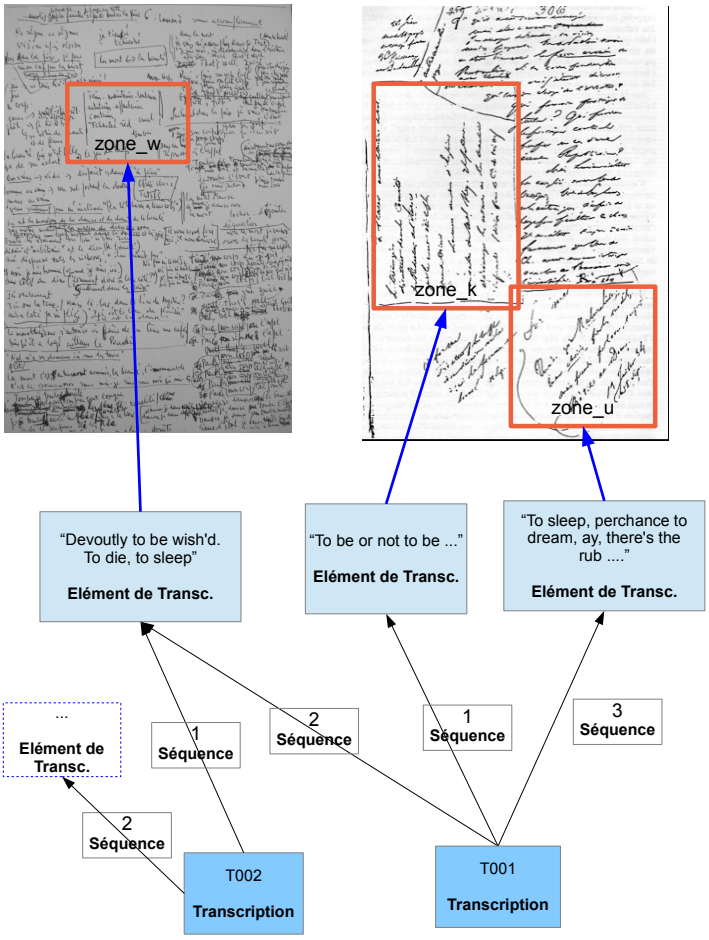


FIGURE 4.1 – *Illustration de deux séquences d'éléments de transcription*

4.1.3 Identification des images de manuscrits (cote)

Les photographies doivent pouvoir être liées à une représentation sémantique de l'objet physique (le manuscrit), ainsi qu'aux annotations, aux transcriptions et autres métadonnées. Comme le modèle de données est basé sur RDF [5] (*Resource Description Framework*), chaque ressource doit être représentée par une URI unique.

Les images du corpus de test fourni pour ce projet sont nommées selon la cote de la Bibliothèque de Genève (BGE). Le nom de fichier est d'ailleurs l'unique moyen de savoir quel manuscrit ou feuillet l'image représente, car aucune autre donnée (ou métadonnée) n'est disponible.

Afin de garder un système cohérent et compréhensible, on utilise la cote (comme il s'agit déjà d'un identifiant unique) pour identifier les fichiers (images) et pour générer les URI des ressources RDF.
N.B. : Le processus d'extraction de la cote et les détails techniques l'accompagnant (convention de nommage, génération d'URI uniques, etc.) sont décrits dans les sections 4.4 et 6.3.

La cote est construite sur plusieurs éléments correspondant à la structure des archives. À Genève (du moins dans le cas qui nous intéresse), les archives possèdent des sections dans lesquelles se trouvent des boîtes. Ces boîtes contiennent les manuscrits (feuilles volantes, cahiers, registres, etc.), normalement préservés dans des enveloppes spéciales (que l'on appelle subdivisions, voir ci-dessous).

5. `http://en.wikipedia.org/wiki/Resource_Description_Framework`

Exemple de cote de la bibliothèque : `Ms. fr. 3951/10, f. 32`

- [**Ms. fr**] la section des **m**anus**c**rits **fr**ançais
- [**3951**] le dossier/la boîte n°**3951** de la section Ms. fr.
- [**10**] la subdivision n°**10** (contenue dans la boîte n°3951)
- [**f. 32**] et le numéro de page/**f**euillet **32**

Exemple de nom de fichier correspondant à la cote ci-dessus : `ms_fr_03951_10_f031v_032.tif`

- [**ms_fr**] la section des **m**anus**c**rits **fr**ançais
- [**03951**] le dossier/la boîte n°**03951** de la section "ms_fr".
- [**10**] la subdivision n°**10** (contenue dans la boîte n°03951)
- [**f031v**] la page 031v (verso de la page 031) du feuillet
- [**032**] et la page 032 (son recto, implicitement).

On constate que le nommage du fichier image reprend globalement la cote de la bibliothèque, malgré quelques petites différences :

- [**03951**] au lieu de [**3951**]
- [**f031v_032**] au lieu de [**f. 32**]
- Tous les séparateurs (espace, point, virgule, *slash*) sont remplacés par des *underscores*
- Toutes les majuscules sont remplacées par des minuscules

Le fichier image est donc nommé de façon univoque et unique. Cependant, l'information sémantique délivrée par la cote n'est en réalité pas fiable à 100%. En effet, un archiviste définit la cote sur la base de son emplacement (section, boîte), puis son organisation dans la boîte (enveloppe, numéro de feuillet/page). Si celui-ci omet un élément (comme le numéro d'une enveloppe) ou s'il commet une erreur, il ne sera pas aisé de modifier la cote *a posteriori* : les archivistes ont pour règle de ne jamais changer une cote existante, car des travaux peuvent y faire référence (articles, livres, etc.).

Il arrive parfois qu'une page manquante soit retrouvée après coup. Dans ce cas, s'il est possible de l'ajouter aux archives existantes, la cote se terminera par un suffixe du type "b" ou "bis". Si par contre, cette page fait partie d'un autre lot d'archives (par ex. : *Ms. Fr* vs. *Arch. Sauss.*), la cote sera complètement différente.

Il est aussi important de noter que le système de cote est en place depuis bien plus longtemps que l'informatisation des bibliothèques. Ce qui peut nous apparaître comme une "erreur" (qu'elle soit de syntaxe ou de conception) par rapport à notre logique moderne, doit être replacé dans son contexte. L'information qu'un archiviste juge pertinente pour l'identification d'un manuscrit en 2013, par rapport à l'un de ses prédécesseurs en 1890, ne peut en aucun cas être la même.

Un problème pour la structure est le cas des "enveloppes". Selon les boîtes, il peut arriver qu'il y ait plusieurs niveaux d'enveloppes (comme des poupées russes), ou qu'il n'y ait pas d'enveloppe du tout. Dès lors, comme le nombre d'enveloppes (niveaux) est variable (entre 0 et N), on définit cette inconnue comme la **subdivision**.

Afin de garder groupés les feuillets se trouvant dans un même conteneur (enveloppe ou sous-enveloppe), il faut un moyen de gérer ce nombre inconnu de niveaux d'enveloppes. Une solution élégante à ce problème consiste à considérer l'ensemble des niveaux, entre la boîte et le feuillet, comme un tout. En effet, si l'on représente les subdivisions sous forme de *strings*, ils sont aisément différenciables les uns des autres. Par exemple, tous les feuillets se trouvant dans la sous-enveloppe 03 de l'enveloppe 01, auront la subdivision **01_03**. Et si la même enveloppe 01 contient également un feuillet qui, pour une raison ou une autre, n'est pas dans une sous-enveloppe, il aura simplement la subdivision **01**. On voit dans cet exemple que l'on peut différencier les subdivisions quel que soit le nombre de niveaux, tout en gardant groupés les feuillets se trouvant dans un même conteneur.

Il faut donc bien considérer la cote comme un identifiant unique pour le feuillet et, jusqu'à un certain point, comme un moyen de déterminer l'emplacement physique d'un document à la bibliothèque (section et boîte). Les subdivisions, pour autant qu'elles existent, ne permettent pas d'obtenir une information détaillée exploitable, mais elles sont utilisables comme un tout (voir ci-dessus). La numérotation des feuillets/pages est correcte dans la majorité des cas, mais il peut y avoir des erreurs, des insertions ou des inversions occasionnelles. Ces informations ont d'ailleurs été confirmées par l'un des archivistes de la BGE à la fin du projet.

4.2 UML et explication du modèle

Comme expliqué dans la section 4.1.3, la cote est l'unique information dont l'on dispose pour identifier les manuscrits. On l'extrait depuis le nom des fichiers image (voir la section 6.3). La cote est décomposée en plusieurs unités et permet de déterminer l'emplacement à la bibliothèque. Le nom de fichier permet de définir le modèle jusqu'à la photo (bibliothèque, section, boîte d'archives, subdivision, surface d'écriture, photo). Les données subséquentes (zone, forme géométrique, annotation, élément de transcription, concept, transcription) sont ajoutées/modifiées selon les actions des utilisateurs.

Le diagramme UML (figure 4.2) contient deux parties. La partie supérieure représente les informations "analogiques" du modèle (de la bibliothèque jusqu'à la surface d'écriture) alors que la partie inférieure représente les informations "numériques" du modèle (de la photographie jusqu'aux annotations ou aux transcriptions).

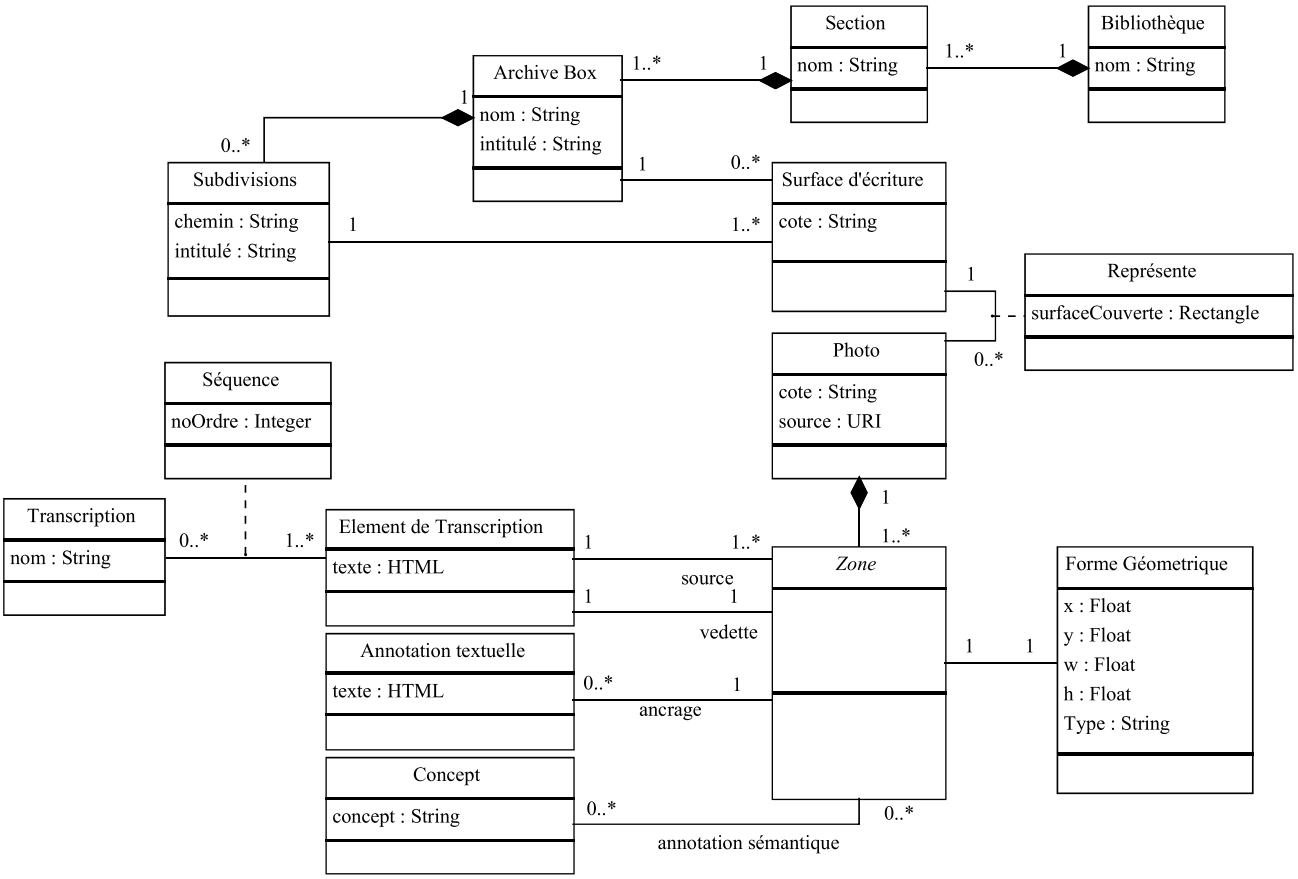


FIGURE 4.2 – *Schéma conceptuel du système des manuscrits*

Ces notions sont représentées en UML. Le diagramme se lit de la manière suivante :

- Une **bibliothèque** contient des **sections**.
- Une **section** contient des **boîtes d'archives**.
- Une **boîte d'archives** peut contenir des **subdivisions** ou des **surfaces d'écriture**.
- Une **subdivision** représente une enveloppe ou plusieurs enveloppes en cascade (en "poupées russes"). La **subdivision** contient des **surfaces d'écriture** (ou **objets physiques**).
- une **surface d'écriture** correspond en général à une page, mais il peut s'agir d'une couverture, d'un côté d'un papier d'emballage, etc.
- Une **photo** représente une partie (**surface couverte**) d'une **surface d'écriture**. Dans la majorité des cas, la **photo** représente l'entier de la **surface d'écriture**.
- une **zone** est une partie d'une **photo**. Chaque **zone** est définie par une **forme géométrique** et par une **annotation**, un **élément de transcription**, ou un **concept**.
- Une **forme géométrique** possède un type (rectangle, etc.) et des valeurs numériques (coordonnées, hauteur, largeur, etc.).
- Une **annotation textuelle** est un texte formaté en HTML.
- Un **élément de transcription** est un texte formaté en HTML qui représente le contenu d'une zone d'une photo.
- Un **concept** est un texte non formaté.
- Les **transcriptions** sont liées à des **éléments de transcription**.

Le terme **surface d'écriture** est utilisé en lieu et place d'**objet physique**, déjà employé dans ce rapport pour décrire le feuillet, par opposition à sa photo. En effet ces deux notions sont redondantes, mais il y une raison historique à cela. Au départ l'**objet physique** devait représenter par exemple un cahier. Les feuillets le composant étaient des **surfaces d'écriture**, dont une partie ou l'ensemble (**surface couverte**) était représenté par une **photo**. Or, il est impossible de déduire l'**objet physique** depuis la cote, car cette information n'existe parfois tout simplement pas, ni dans le nom de fichier, ni dans le système de cote de la bibliothèque. Comme le terme **objet physique** est très parlant, il a été gardé pour désigner la **surface d'écriture**, par abus de langage.

4.3 RDFS

Le RDFS (RDF Schema) permet de définir des classes et des propriétés personnalisées et adaptées à l'application. Il s'agit également une manière de poser le vocabulaire utilisé dans le modèle de données.

Lorsque le triplestore gère le système d'implication (*entailment*), cela lui permet de faire des **déductions**. Dans une base de connaissance, cela est important pour mettre en rapport différentes données qui ne sont pas directement liées, mais entre lesquelles un lien peut être déduit.

```
ex:RaymondDeSaussure     ex:filsDe   ex:FerdinandDeSaussure .
ex:filsDe rdfs:domain ex:Homme .
ex:filsDe rdfs:range ex:Parent .
```

Cela permet de déduire que pour le prédicat `ex:filsDe`, les sujets sont toujours du type `ex:Homme`, et que les objets sont toujours du type `ex:Parent`. Si le triplestore prend en compte le RDFS, une requête SPARQL cherchant tous les parents (tous les éléments de type `ex:Parent`) retournera `ex:FerdinandDeSaussure` dans la liste des résultats. Pourtant `ex:FerdinandDeSaussure` n'est pas déclaré comme étant une instance de la classe `ex:Parent` ; ce lien a été déduit.

On peut atteindre le même résultat sans utiliser RDFS, simplement en étant plus exhaustif lors de la déclaration (la déclaration du type est cependant nécessaire à chaque ajout) :

```
ex:RaymondDeSaussure     ex:filsDe   ex:FerdinandDeSaussure .
ex:FerdinandDeSaussure   rdf:type    ex:Parent .
ex:RaymondDeSaussure     rdf:type    ex:Homme .
```

OWL est une extension de RDFS. Elle offre bien plus de possibilités que RDFS. Malheureusement, les triplestores n'intègrent que peu (ou pas) ces deux langages. Il n'est donc pas raisonnable de compter sur ces langages et leurs possibilités.

Le listing 4.1 contient la déclaration du schéma RDFS correspondant au modèle utilisé dans ce travail. On retrouve le "vocabulaire" utilisé dans le modèle de données.

```
@prefix rdfs: <http://www.w3.org/2000/01/rdf-schema#> .
@prefix rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#> .
@prefix saussure: <http://saussure.com/ressource/>.
@prefix p: <http://saussure.com/property/>.

### Classes
saussure:Section              rdf:type rdfs:Class .
saussure:ArchiveBox           rdf:type rdfs:Class .
saussure:Subdivisions         rdf:type rdfs:Class .
saussure:Surface_d_ecriture   rdf:type rdfs:Class .
saussure:Photo                rdf:type rdfs:Class .
saussure:Zone                 rdf:type rdfs:Class .
saussure:FormeGeometrique     rdf:type rdfs:Class .
saussure:Annotation           rdf:type rdfs:Class .
saussure:TranscriptionElement rdf:type rdfs:Class .
saussure:Transcription        rdf:type rdfs:Class .
saussure:ConceptualAnnotation rdf:type rdfs:Class .

### Properties
p:hasArchiveBox       rdf:type rdf:Property .
p:hasSubdivisions     rdf:type rdf:Property .
p:hasSurfaceEcriture  rdf:type rdf:Property .
p:hasCote             rdf:type rdf:Property .
p:hasPhoto            rdf:type rdf:Property .
p:hasSource           rdf:type rdf:Property .
p:hasFilename         rdf:type rdf:Property .
p:contient            rdf:type rdf:Property .
p:hasAnnotation       rdf:type rdf:Property .
p:hasForme            rdf:type rdf:Property .
p:texte               rdf:type rdf:Property .
p:rawText             rdf:type rdf:Property .
p:hasCreatorName      rdf:type rdf:Property .
p:hasCreationDate     rdf:type rdf:Property .
p:hasEditorName       rdf:type rdf:Property .
p:hasEditionDate      rdf:type rdf:Property .
p:hasX                rdf:type rdf:Property .
p:hasY                rdf:type rdf:Property .
p:hasHeigth           rdf:type rdf:Property .
p:hasWidth            rdf:type rdf:Property .
```

Listing 4.1 – Déclaration RDFS

L'intérêt principal de RDFS est de pouvoir faire des déductions dans des graphes dont la structure est complexe et dont les nœuds ne peuvent pas être aisément liés. Or, lorsqu'on analyse le diagramme UML du modèle (figure 4.2), on se rend compte que les données sont organisées de façon hiérarchique. En effet, le système de classement de la bibliothèque utilise clairement une structure en arbre et le reste du modèle également (photographies, zones, annotations, etc.).

L'application développée n'a pas besoin de pouvoir faire des déductions. En effet, le modèle prend en compte les différents degrés nécessaires à l'extraction de l'information. Par exemple, la

cote est découpée en section, boîte, subdivision et feuillet. Cela permet d'extraire, par exemple, toutes les boîtes d'une section ou tous les feuillets d'une boîte.

Les déductions ne sont pas réellement importantes non plus, dans la mesure où les demandes d'ajout, de modification et de suppression passent par le service frontal. Elles sont ensuite traitées pour garantir l'unicité et la cohérence des données dans le triplestore. A ce moment, un sous-graphe est inséré (INSERT), effacé (DELETE) ou modifié (DELETE/INSERT). Ce sous-graphe est construit grâce à une fonction générique qui ajoute systématiquement les détails d'appartenance à une classe. Il n'est donc plus nécessaire de les déduire si l'information du graphe construit est déjà suffisamment riche.

4.4 RDF et génération d'URI uniques

Les données du projet sont au format RDF et stockées dans un triplestore. Les ressources sont donc identifiées par des URI. Contrairement à une base de données relationnelle, où la génération d'identifiants uniques fait partie intégrante du système de gestion de base de données, la génération d'URI uniques pour RDF est à la charge du développeur. Il est donc important d'avoir un système cohérent et robuste pour assurer leur unicité.

On définit deux préfixes. Le préfixe **p** est utilisé pour décrire les propriétés (prédicats). Le préfixe **saussure** est utilisé pour décrire les ressources (sujets/objets) :

```
PREFIX p:<http://saussure.com/property/>
PREFIX saussure:<http://saussure.com/ressource/>
```

N.B. : Le nom de domaine **saussure.com** existe réellement, et n'a aucun rapport avec ce projet. Il est utilisé à titre illustratif dans les tests et dans les exemples de ce travail, car il est court et intuitif. Le nom de domaine qui sera réellement utilisé lors de la mise en production n'est pas encore défini.

La stratégie utilisée dans ce projet est simple, mais efficace. Les URI sont générées depuis la cote extraite du nom du fichier image (voir la section 6.3). On utilise une approche "cumulative", c'est-à-dire que les URI sont créés dans l'ordre du modèle ; chaque nouvelle URI reprend la précédente et lui ajoute un degré d'information (voir l'exemple ci-dessous). Ceci est possible, car on se base sur la cote dont la structure est hiérarchique et structurée (par ex. : il n'y a pas de risque que deux manuscrits appartiennent simultanément à deux sections ou à deux boîtes).

La construction des URI est effectuée lors de l'ajout d'une image et de l'ajout d'annotations, d'éléments de transcription ou d'annotations conceptuelles. Les systèmes de contrôle et de vérification mis en place sont décrits dans les sections 6.3 et 6.4.

L'exemple suivant illustre les différentes URI des ressources et des propriétés nécessaires à la représentation de l'image : **ms__fr__03965__03__f043v__044**

```
saussure:ms_fr rdf:type saussure:Section ;
        p:hasArchiveBox saussure:ms_fr_03951 .
saussure:ms_fr_03951 rdf:type saussure:ArchiveBox ;
        p:hasSubdivisions saussure:ms_fr_03951_10 .
saussure:ms_fr_03951_10 rdf:type saussure:Subdivisions ;
        p:hasSurfaceEcriture saussure:ms_fr_03951_10_f028v_029 .
saussure:ms_fr_03951_10_f028v_029 rdf:type saussure:Surface_d_ecriture ;
        p:hasCote "ms_fr_03951_10_f028v_029";
        p:hasPhoto saussure:ms_fr_03951_10_f028v_029-DOT-jp2 .
saussure:ms_fr_03951_10_f028v_029-DOT-jp2 rdf:type saussure:Photo ;
        p:hasCote "ms_fr_03951_10_f028v_029" ;
        p:hasSource "http://saussure.unige.ch/bge/manuscript/ms_fr_03951_10_f028v_029.jp2" ;
        p:hasFilename "ms_fr_03951_10_f028v_029.jp2" .
```

Listing 4.2 – Ajout de l'image ms_fr_03951_10_f028v_029.jp2

Avec RDF/SPARQL, il ne porte pas à conséquence d'ajouter plusieurs fois le même triplet, même si celui-ci existe déjà dans le triplestore, car le sujet, le prédicat et l'objet sont identifiés de manière unique grâce aux URI.
Dans les cas où l'on maîtrise la structure de données, cela permet d'économiser des opérations de contrôle ; on peut donc se permettre d'ajouter systématiquement une information complète, sans devoir vérifier l'existence préalable de certains triplets. Dans l'exemple ci-dessus, si le manuscrit (**Surface_d_ecriture**) n'existe pas dans le triplestore, alors la section, la boîte et la subdivision sont toujours ajoutées, sans autre forme de contrôle.

L'exemple suivant illustre la création d'une annotation. Lors de l'ajout d'une annotation, les URI sont générées depuis la **Photo** de l'exemple précédent :

saussure:ms_fr_03951_10_f028v_029-DOT-jp2 rdf:type saussure:Photo .

```
INSERT DATA {
   GRAPH <file://test_001_2013-04-08.xml> {
      # INSERT ZONE dans Photo
      saussure:ms_fr_03951_10_f028v_029-DOT-jp2 rdf:type saussure:Photo ;
      p:contient saussure:ms_fr_03951_10_f028v_029_Z_001 .
      # CREATE ZONE + annotation + shape
      saussure:ms_fr_03951_10_f028v_029_Z_001 rdf:type saussure:Zone ;
         p:hasAnnotation saussure:ms_fr_03951_10_f028v_029_Z_001_annot_001 ;
         p:hasForme saussure:ms_fr_03951_10_f028v_029_Z_001_Shape_001 .

      saussure:ms_fr_03951_10_f028v_029_Z_001_annot_001 rdf:type saussure:Annotation ;
         p:texte """<p>Ceci est une annotation</p>""" ;
         p:rawText """Ceci est une annotation""" ;
         p:hasCreatorName   """massimo""" ;
         p:hasCreationDate   """2013-07-21_16-04-04""" ;
         p:hasEditorName   """massimo""" ;
         p:hasEditionDate   """2013-07-21_16-04-04""" .

      saussure:ms_fr_03951_10_f028v_029_Z_001_Shape_001 rdf:type saussure:FormeGeometrique ;
         p:hasX "0.11674008810573";
         p:hasY "0.074626865671642";
         p:hasHeigth "0.059701492537313";
         p:hasWidth "0.19383259911894" .
   }
}
```

Listing 4.3 – Ajout d'une annotation au manuscrit ms_fr_03951_10_f028v_029

Grâce à la structure de données et la génération d'URI effectuée en utilisant des identifiants verbeux, on voit dans ces exemples que chaque ressource est clairement identifiée de manière unique. On remarque également que chaque niveau de l'arborescence cumule les informations de son parent :

- saussure:ms_fr
- saussure:ms_fr_03951
- saussure:ms_fr_03951_10
- saussure:ms_fr_03951_10_f028v_029
- saussure:ms_fr_03951_10_f028v_029-DOT-jp2
- saussure:ms_fr_03951_10_f028v_029_Z_001
- saussure:ms_fr_03951_10_f028v_029_Z_001_annot_001
- saussure:ms_fr_03951_10_f028v_029_Z_001_Shape_001

Même si la probabilité est faible que deux ressources aient le même identifiant, elle n'est statistiquement pas nulle. Par exemple, si à la BGE il existe une boîte 3965 dans la section des "archives de Saussure" et une boîte 3965 dans la section des "manuscrits français", alors ces deux boîtes auront respectivement les URI **saussure:arch_saussure_03965** et **saussure:ms_fr_03965**, plutôt qu'une URI ambiguë comme **saussure:03965**. Si cette

dernière URI était utilisée, cette représentation serait erronée. En effet, les sections **saussure:arch_saussure** et **saussure:ms_fr** auraient toutes deux un prédicat **p:hasArchiveBox** pointant le même objet **saussure:03965**. Il serait donc impossible de savoir à laquelle des deux boîtes 3965 les manuscrits sont rattachés. Il est donc indispensable que le système prenne en compte ceci dans les procédures d'ajout.

Les URI des zones, des annotations et des formes contiennent des identifiants numériques incrémentiels (compteurs). Leur génération nécessite une vérification préalable pour s'assurer qu'ils n'existent pas déjà dans le triplestore.

Chapitre 5

Interfaces utilisateurs

L'analyse des besoins des utilisateurs a permis de déterminer que ceux-ci souhaitent pouvoir visualiser l'image et interagir avec elle. Très rapidement, les utilisateurs du projet ont souhaité que l'interface soit centrée sur l'image du manuscrit. Outre le zoom et le *panning*[1], ils doivent également pouvoir dessiner des zones afin d'y lier du texte (annotation). Il doit aussi être possible de visualiser les autres feuillets en lien avec le manuscrit consulté (par ex. : tous les feuillets d'un cahier). Avant d'aboutir à une version finale concluante, plusieurs prototypes d'interface ont été conçus. Ils sont sommairement décrits ici afin de suivre les étapes de l'évolution du prototype jusqu'à sa version finale.

5.1 Premiers prototypes

5.1.1 Premier prototype

Le premier prototype se base sur deux modes de visualisation :

- La vue **objet physique** (voir la Table 5.1 et la figure 5.1) permet de visualiser simultanément plusieurs images représentant un tout (par ex. : un cahier), afin d'offrir une représentation visuelle la plus proche possible de l'objet réel. Cette représentation nécessite de pouvoir connaître le rapport de position (précédent/suivant, recto/verso) entre deux documents.
- La vue d'**approfondissement** (ou **triptyque**) (voir la Table 5.2 et la figure 5.2) permet de visualiser (et d'éditer) des informations détaillées d'un manuscrit en affichant le "triptyque"[2] manuscrit/annotations/transcription :
  - Le "triptyque" est un affichage qui se découpe en trois "panneaux" (ou zones) : une zone pour le manuscrit, une zone pour les annotations et une zone pour la transcription.
  - Une seule image est utilisée dans le "triptyque", par opposition à la vue "objet physique" qui en affiche plusieurs.

	Page 1
Page 2	Page 3
Page 4	

Table 5.1 – Schéma de la vue "objet physique"

1. `http://hea-www.harvard.edu/RD/saotng/panzoom.html`
2. Utilisation de l'analogie du triptyque `http://fr.wikipedia.org/wiki/Triptyque_(Beaux-arts)`

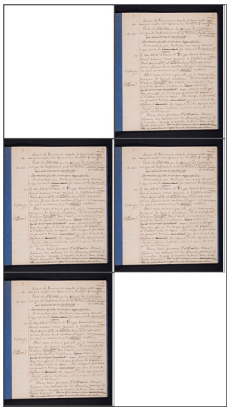


FIGURE 5.1 – *Capture d'écran d'un "objet physique" avec des thumbnails*

Annotations	Manuscrit	Transcription

TABLE 5.2 – Schéma de la vue "triptyque"

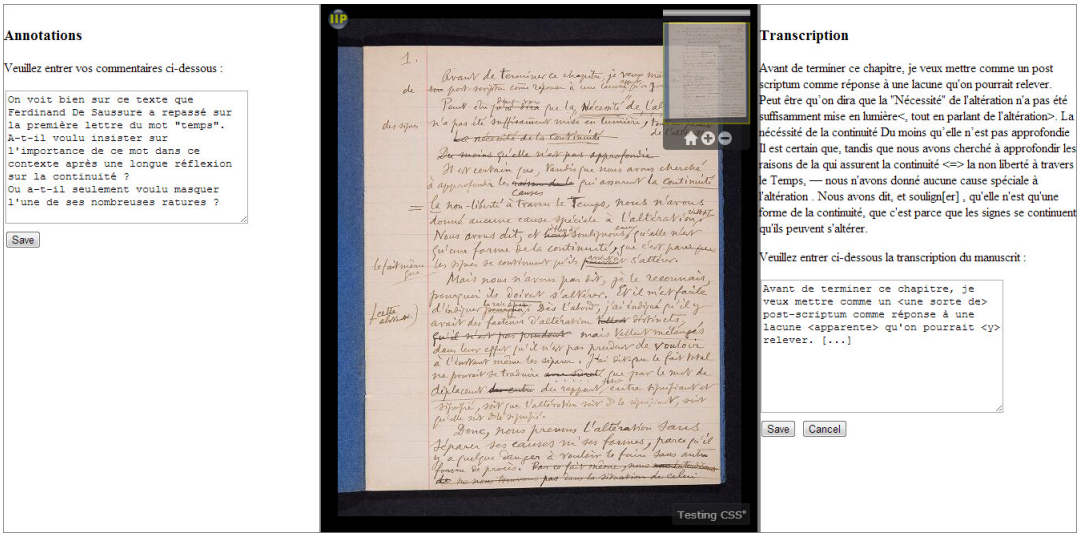


FIGURE 5.2 – *Capture d'écran d'un "triptyque"*

La première approche pour la vue "objet physique" consistait à afficher un viewer par image. La page web était partagée entre autant de viewers que nécessaire. La pratique a montré que l'emploi d'un viewer par image ne s'avère pas très concluant. En effet, le fait de pouvoir zoomer indépendamment sur chaque image rend rapidement cette représentation complètement confuse. De plus, comme tous les viewers doivent se partager l'espace disponible, la zone que représente chaque viewer est passablement réduite et rend la lecture du manuscrit peu agréable.

La seconde approche était d'utiliser des *thumbnails* cliquables amenant chacun à la vue d'approfondissement correspondant au feuillet cliqué. L'utilisation de *thumbnails* permet d'avoir une bonne idée de la manière dont les feuillets sont organisés dans l'objet physique, tout en étant beaucoup plus agréable à visualiser, même si l'image est alors trop petite pour la lecture.

Cette méthode présente le désavantage de devoir construire avec des *thumbnails* une représentation cohérente pour chaque objet. Comme de Saussure écrivait sur tout et n'importe quoi (cahier, faire-part, papier d'emballage, etc.) et avec une logique bien à lui, il n'est pas forcément évident de définir un ordre de visualisation pour tous les manuscrits.

En ce qui concerne la vue d'approfondissement (triptyque), elle permet de voir les trois éléments principaux en un coup d'œil, sur une seule page. Cependant, même sur un écran large, la place que prennent les annotations et les transcriptions n'est pas idéale pour la lecture du manuscrit.

Ce premier prototype propose une ébauche d'interface quelque peu laborieuse qui n'a pas été retenue. La vue "objet physique" n'est pas indispensable, car l'information la plus pertinente qui en ressort est l'ordre des feuillets entre eux. Cet ordre peut être représenté par une simple suite (en ligne ou en colonne) que l'on peut alors facilement intégrer dans la vue d'approfondissement.

5.1.2 Second prototype

La seconde version du prototype propose de garder l'image et les transcriptions côte à côte, mais de mettre les annotations en dessous, permettant de gagner de l'espace en largeur au détriment de l'espace en hauteur (voir la Table 5.3).

La figure 5.3 est un *mock* réalisé avec Lumzy [3]. Quelques additions sont proposées, comme par exemple la possibilité de choisir les versions des transcriptions dans une liste, ou encore la navigation entre les feuillets du même objet avec des *thumbnails* (ceci correspond à l'intégration dans une seule interface de la vue "objet physique" du prototype précédent). Les annotations (partie inférieure) suggèrent la possibilité d'afficher les annotations une par une, et de les filtrer.

Manuscrit	Transcription
Annotations	

TABLE 5.3 – Schéma de la vue du deuxième prototype

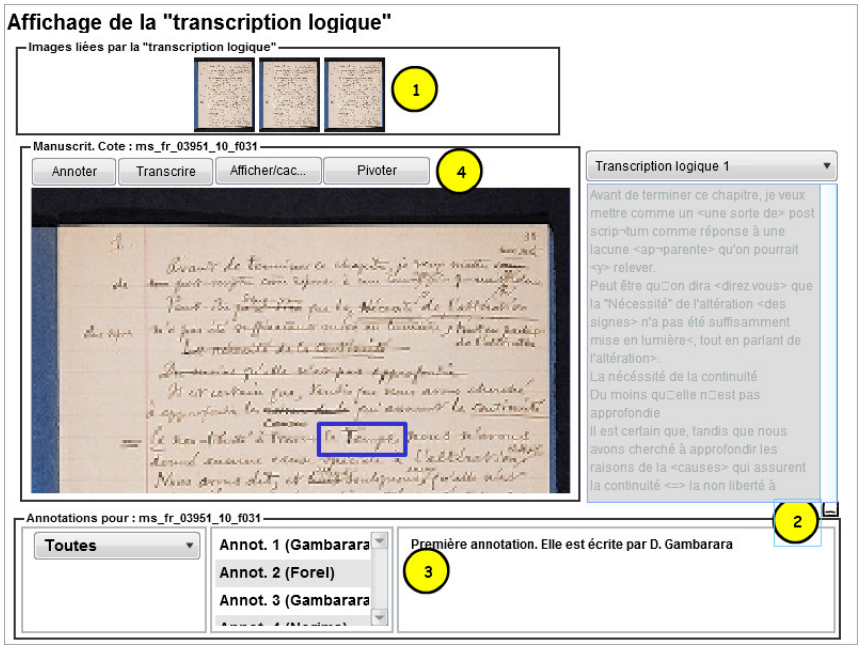


FIGURE 5.3 – *Capture d'écran du mock*

Ce *mock* n'a pas été implémenté tel quel. La navigation placée au-dessus de l'image du manuscrit n'est pas très intuitive : les utilisateurs préféraient la voir en dessous de l'image ou sur la gauche de l'image (verticalement). Ces modifications rendaient cette interface peu intuitive et une impression d'"encombrement" s'en dégageait.

3. `http://www.lumzy.com`

5.1.3 Troisième prototype

Le troisième prototype (voir la figure 5.4) se base sur le projet Lincolnnus/Image-Viewer (voir section 3.4.4.4). Par rapport au *mock* du prototype précédent, la navigation est placée sous l'image et la zone d'annotation est réduite à une zone dans le coin inférieur droit. Cela permet un gain d'espace important en hauteur, en plaçant la navigation et les annotations côte à côte.

Ce prototype est plus abouti que les autres, car il commence à réellement intégrer une bonne partie des fonctionnalités. Il propose aussi une première version rudimentaire de la bannière en haut de la page.

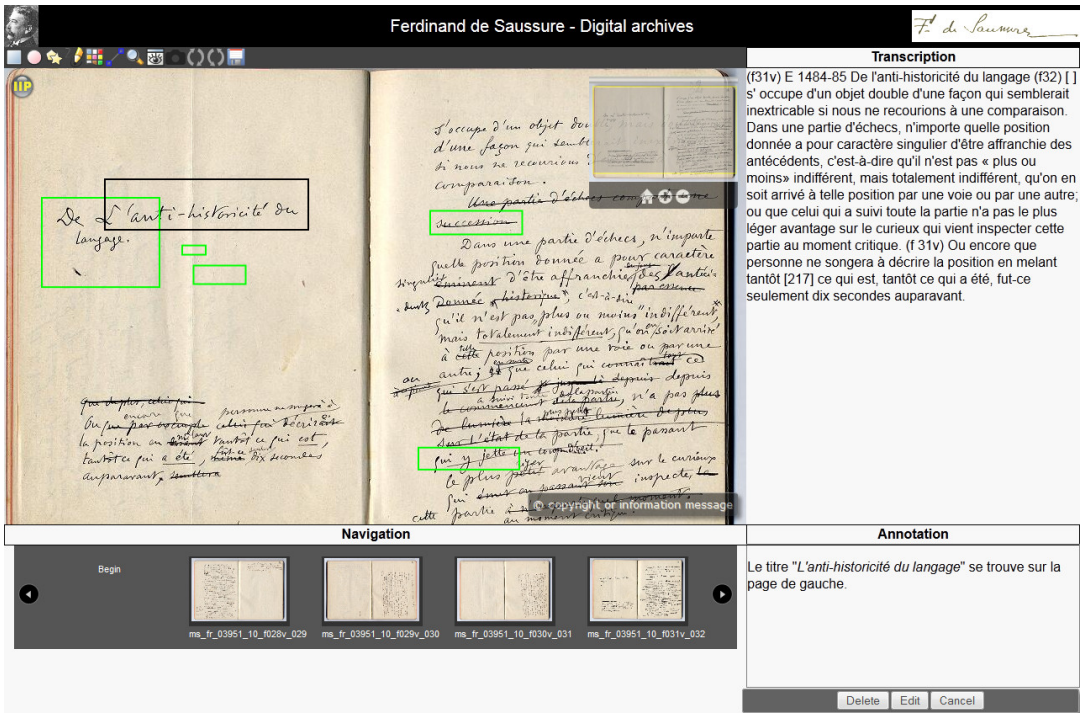


FIGURE 5.4 – *Capture d'écran du troisième prototype*

Malgré un gros progrès en terme d'agencement de l'espace par rapport au prototype précédent (*mock*), ce troisième prototype n'a pas été retenu : une place importante reste perdue en hauteur, car la zone de navigation est plus petite que la zone d'annotation. Le prototype final propose une alternative qui optimise l'espace et le réorganise lors de certaines interactions. Il faut aussi noter qu'il a malheureusement été nécessaire d'abandonner le projet Lincolnnus/Image-Viewer et de réécrire tout le module d'annotation (voir 6.7.3.1).

5.2 Prototype final

Le prototype final améliore l'interface utilisateur de l'application client dont on a vu les premiers prototypes. Il ajoute aussi une deuxième partie au site web : l'accès aux manuscrits grâce à différentes méthodes.

Sur le site, quelle que soit la page consultée, la bannière est toujours présente. Elle permet de revenir à la page d'accueil en cliquant sur le portrait ou la signature de Ferdinand de Saussure, ainsi que de changer la langue (français ou anglais) en cliquant sur le drapeau correspondant. La bannière permet également à l'utilisateur de s'authentifier (figure 5.5), ou de vérifier son statut de connexion (figure 5.6).

FIGURE 5.5 – *La bannière lorsque l'utilisateur n'est pas connecté*

FIGURE 5.6 – *La bannière lorsque l'utilisateur est connecté*

5.2.1 Application client

La version finale du prototype (voir la figure 5.7) propose une interface reprenant certaines idées des prototypes précédents, mais en les organisant de façon plus ergonomique. On décrit ici cette nouvelle interface utilisateur et les fonctionnalités qu'elle offre. Contrairement aux prototypes précédents, toutes les fonctionnalités décrites ici sont implémentées et fonctionnelles.

L'interface doit être centrée sur l'image du manuscrit. La visualisation d'une image est accompagnée de l'affichage de métadonnées (cote, annotations, etc.). Ces métadonnées sont chargées depuis (ou sauvées dans) le triplestore. On a pu apprendre des prototypes précédents que l'image doit prendre beaucoup d'espace, et que le manuscrit et sa transcription doivent être côte à côte pour permettre une lecture agréable. Il est donc important d'optimiser l'organisation des éléments sur la page et de trouver un compromis pour fournir un maximum de fonctionnalités dans l'espace restant. Le meilleur moyen d'y arriver est d'avoir une interface aussi intuitive que possible.

Les améliorations par rapport aux prototypes précédents font encore gagner de l'espace. La première est la transformation de la navigation. Elle est placée sur la gauche du viewer et, pour la première fois, à la verticale. On revient à une vue en triptyque, sauf qu'ici la partie contenant les annotations est remplacée par la navigation. La seconde amélioration de taille se situe au niveau des annotations, qui n'occupent de l'espace sur la page que lors de l'édition (ajout/modification). Le contenu de l'annotation est affiché lorsqu'on passe la souris sur la zone (voir la figure 5.14), et un éditeur s'ouvre seulement lors de l'édition (voir la figure 5.23).

L'une des améliorations, en termes de fonctionnalités cette fois, est l'ajout des "modes". Ceux-ci sont représentés par les trois boutons sous l'image. Ils permettent avec la même interface d'afficher, de créer et d'éditer des "annotations textuelles", des "éléments de transcription", et des "annotations conceptuelles".

L'interface principale (figure 5.7) se décompose en plusieurs zones : la bannière en haut (qui, comme on l'a déjà vu, est toujours présente), et trois zones principales (gauche, centrale et droite).

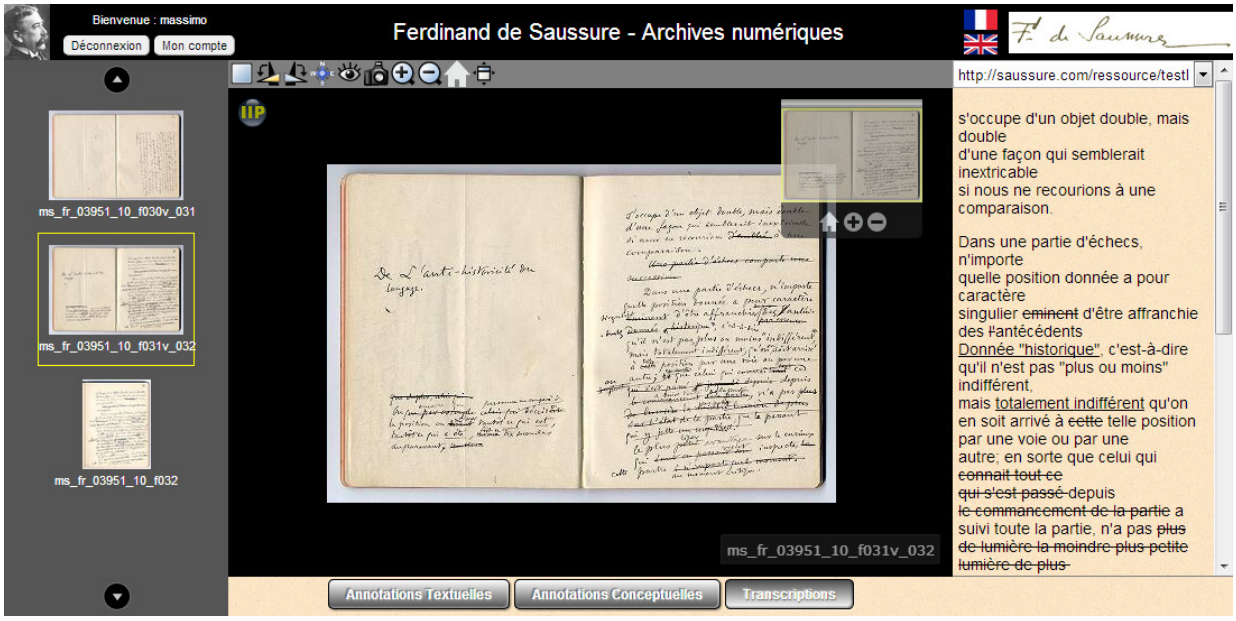


FIGURE 5.7 – *Capture d'écran du prototype final*

Dans le panneau de gauche se trouve la navigation par *thumbnails*, ou carrousel, qui permet de naviguer entre les feuillets du même objet (figure 5.8). Les flèches (en haut, en bas) permettent de faire défiler les *thumbnails*. Lorsqu'on clique sur un *thumbnail*, l'image au centre est rechargée, ainsi que les annotations ou les transcriptions (selon le mode dans lequel on se trouve).

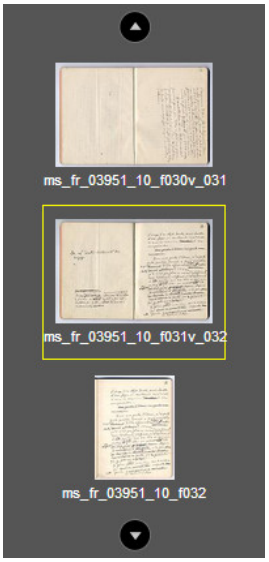


FIGURE 5.8 – *Navigation par thumbnails (carrousel)*

Dans le panneau de droite se trouvent les transcriptions du manuscrit courant, pour autant qu'il ait été transcrit (figure 5.9). La liste déroulante en haut du panneau de droite permet de choisir entre les transcriptions lorsqu'il en existe plusieurs (voir la section 4.1.2). La transcription sélectionnée est alors affichée en dessous, côte à côte avec l'image.
N.B. : Les éléments de transcription peuvent exister, mais le panneau de droite n'affichera aucun contenu tant qu'un ordre de lecture n'aura pas été défini et validé par un expert.

Le panneau central contient le viewer IIPMooViewer, c'est-à-dire l'image zoomable du manuscrit. Une barre d'outils se trouve au-dessus du viewer (entre l'image et la bannière).

Elle permet de réaliser des tâches en rapport avec le viewer. En dessous de l'image se trouvent trois boutons qui permettent de basculer d'un mode à un autre pour le manuscrit courant.

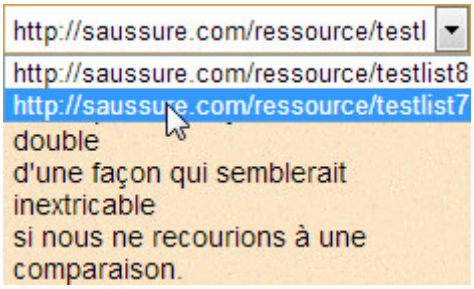


FIGURE 5.9 – *Liste des différentes transcriptions d'un manuscrit*

La barre d'outils (*extended-toolbar*) a été développée dans ce projet sous la forme d'une extension pour IIPMooViewer (voir section 6.7.3). Elle étend ses fonctionnalités en permettant un accès facile et visuel à différentes actions (figure 5.10). La création d'annotations, d'éléments de transcription ou d'annotations conceptuelles nécessite que l'utilisateur soit authentifié. Le bouton permettant la création est alors grisé lorsque l'utilisateur n'est pas connecté (figure 5.11).

FIGURE 5.10 – *La barre d'outils (extended-toolbar) lorsque l'utilisateur est connecté*

FIGURE 5.11 – *La barre d'outils (extended-toolbar) lorsque l'utilisateur n'est pas connecté*

Les boutons de la barre d'outils permettent d'agir sur l'interface du viewer ou sur l'image. Voilà la liste des actions qu'ils exécutent (la numérotation correspond à celle de la figure 5.12) :

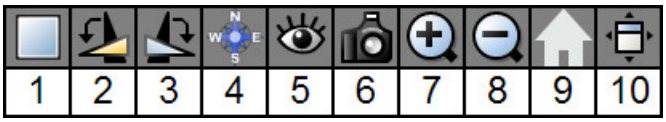
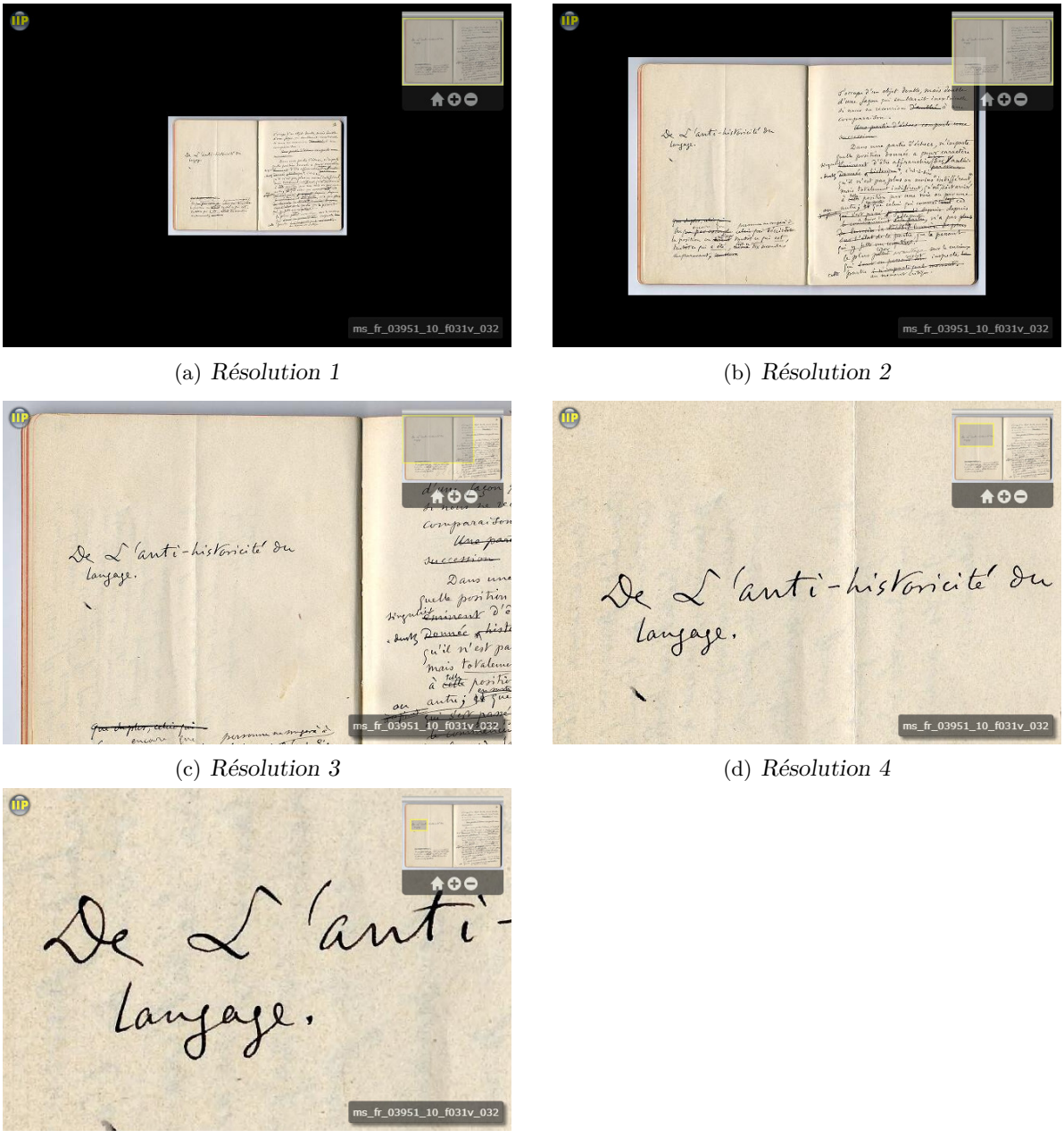


FIGURE 5.12 – *Les boutons de la barre d'outils (extended-toolbar)*

1. Dessiner une zone et y lier une information. Selon le mode sélectionné, il peut s'agir d'une annotation, d'un élément de transcription ou d'une annotation conceptuelle.
2. Pivoter l'image dans le sens antihoraire (+90˚)
3. Pivoter l'image dans le sens horaire (-90˚)
4. Montrer/cacher la fenêtre de navigation
5. Montrer/cacher les zones sur l'image (annotation, etc.)
6. Exporter une image de la vue courante
7. Zoom avant
8. Zoom arrière
9. Réinitialiser la vue
10. Passer en mode plein écran

Le viewer permet de zoomer avec les boutons +/- du clavier, de la mini-map et de la barre d'outils. Il permet également de zoomer en double-cliquant sur l'image, ou encore en utilisant la molette de la souris. La figure 5.13 permet de voir ce qu'affiche le viewer à tous les niveaux de zoom pour ce manuscrit (figure 5.13).



(a) *Résolution 1*

(b) *Résolution 2*

(c) *Résolution 3*

(d) *Résolution 4*

(e) *Résolution 5*

FIGURE 5.13 – *Les différentes résolutions (niveaux de zoom) d'un feuillet*

Lorsque la souris est placée sur l'image, les zones existantes apparaissent en superposition (figure 5.14). Selon le mode choisi, les zones représentent soit des annotations, des éléments de transcription ou des annotations conceptuelles. En effet les boutons qui se trouvent sous le viewer permettent de basculer entre les modes **Annotations Textuelles**, **Annotations Conceptuelles** et **Transcriptions**. Un seul mode peut être sélectionné à la fois, et le bouton du mode en question apparaît alors comme enfoncé (voir les figures 5.15, 5.16, et 5.17).

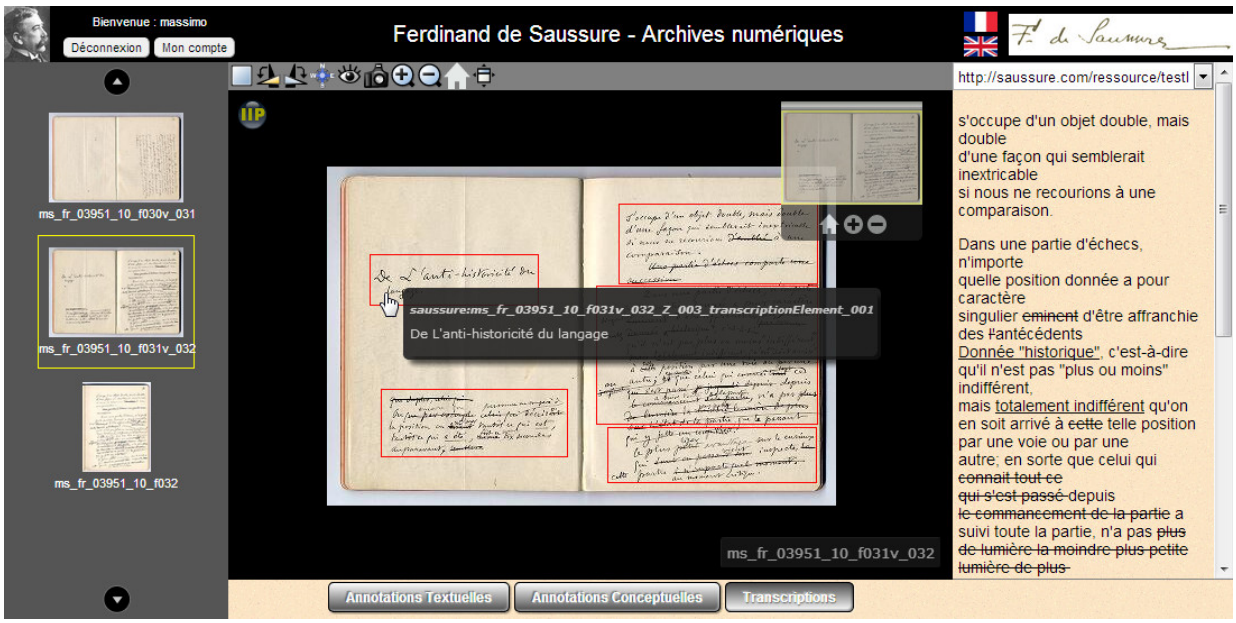


FIGURE 5.14 – *Capture d'écran du prototype final avec des zones visibles*

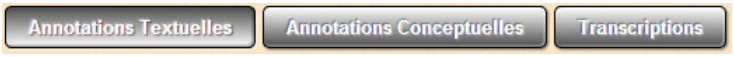


FIGURE 5.15 – *Les boutons des modes, ici en mode Annotations Textuelles*

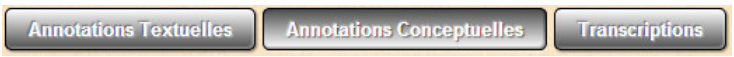


FIGURE 5.16 – *Les boutons des modes, ici en mode Annotations Conceptuelles*

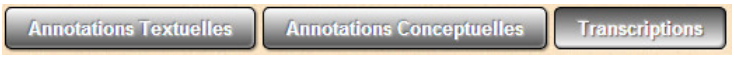


FIGURE 5.17 – *Les boutons des modes, ici en mode Transcriptions*

Les trois modes fonctionnent de la même façon. Par exemple, si le mode **Annotations Textuelles** est sélectionné, tout ce qui est lié aux zones sera du type **Annotations Textuelles**. C'est-à-dire que les zones sur l'image seront de ce type, et la création d'une nouvelle zone sera automatiquement de ce type. Lorsqu'on bascule d'un mode à l'autre, les zones existantes sont rechargées depuis le triplestore.

En plus du bouton qui apparaît comme enfoncé, le mode est également indiqué par la couleur des zones :

- Bleu pour les **Annotations Textuelles** (figure 5.18)
- Violet pour les **Annotations Conceptuelles** (figure 5.19)
- Rouge pour les **Transcriptions** (figure 5.20)

La création d'une zone se fait en cliquant sur le bouton "dessiner une zone" (bouton 1 de la figure 5.12). Le clic sur l'image définit le point de départ et en déplaçant la souris (sans relâcher le clic) une zone jaune se dessine sur l'image. Lorsqu'on relâche le bouton de la souris, la zone est définie et l'éditeur (ou la liste de concepts) apparaît en dessous du viewer (figure 5.21 et 5.22). À ce stade, la zone peut encore être modifiée, et le texte peut être saisi dans l'éditeur (ou le concept peut être sélectionné dans la liste déroulante). Lorsque le bouton **ok** est cliqué, la zone et le texte (ou le concept) sont sauvées dans le triplestore. Si le bouton **cancel** est cliqué, la zone et le texte (ou le concept) sont effacées localement et rien n'est envoyé au triplestore. Le bouton **delete** a le même effet dans ce contexte.

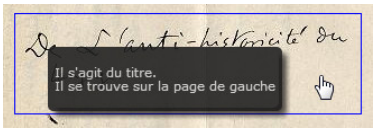


FIGURE 5.18 – *Exemple d'une annotation (cadre bleu)*

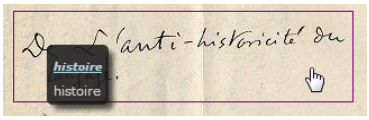


FIGURE 5.19 – *Exemple d'une annotation conceptuelle (cadre violet)*

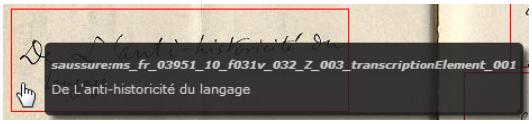


FIGURE 5.20 – *Exemple d'un élément de transcription (cadre rouge)*

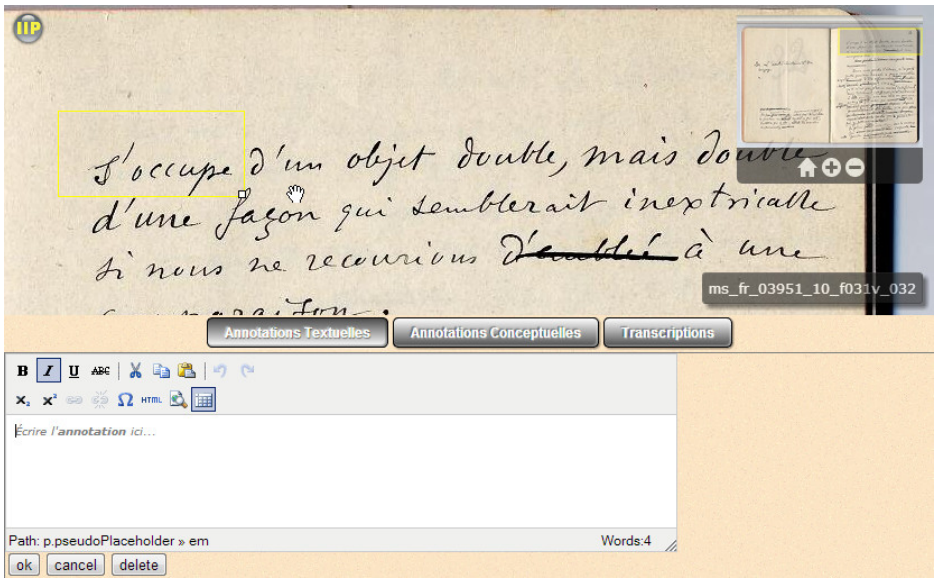


FIGURE 5.21 – *Création d'une annotation autour de "s'occupe"*

Pour modifier l'information liée à une zone, ou pour modifier la zone elle-même (la redimensionner ou la déplacer), ou encore pour effacer définitivement la zone et l'information qui lui est liée, il suffit de double-cliquer sur la zone. Cela fait apparaître l'éditeur, ou la liste des concepts, comme lors de la création d'une zone (figures 5.23 et 5.24). Il est possible de modifier l'information (le texte ou le concept), ainsi que la zone. Le bouton **ok** sauve les modifications et les informations sont mise à jour dans le triplestore. Le bouton **cancel** annule les modifications effectuées. Dans ce contexte, le bouton **delete** efface définitivement du triplestore la zone et l'information qui lui est liée.

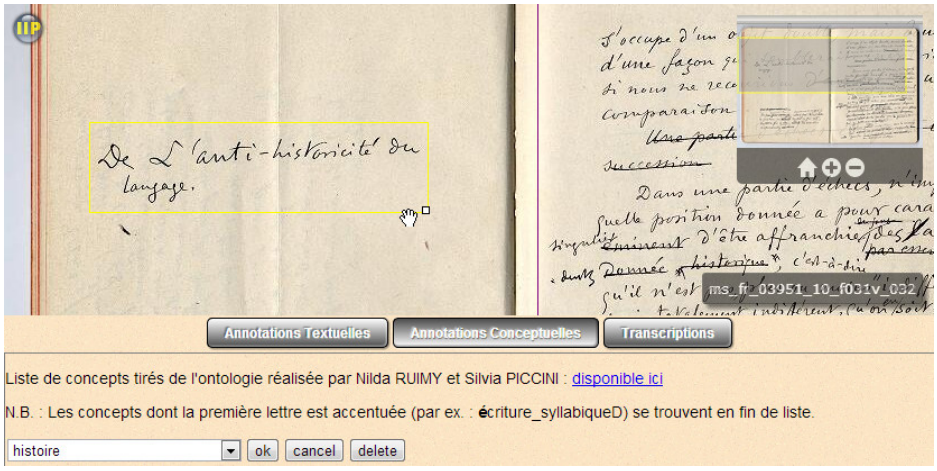


FIGURE 5.22 – *Création d'une annotation conceptuelle*

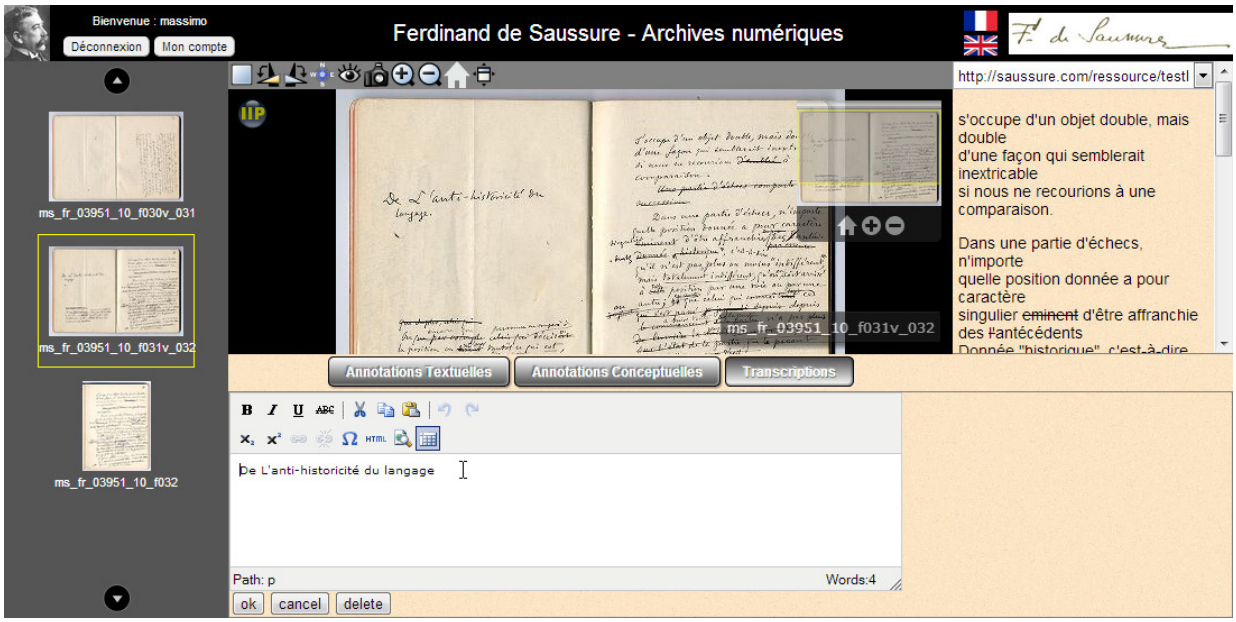


FIGURE 5.23 – *Capture d'écran du prototype final lors de l'édition d'un élément de transcription*

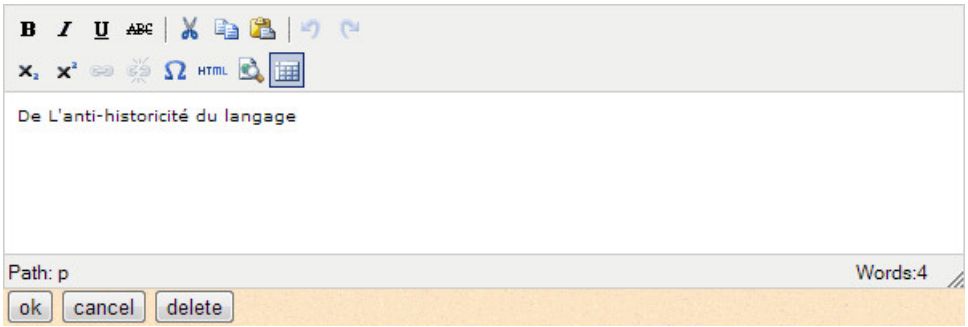


FIGURE 5.24 – *L'éditeur HTML (WYSIWYG) TinyMCE*

5.2.2 Accéder aux manuscrits : recherche et navigation

La section précédente détaillait l'interface utilisateur avec le manuscrit au centre. Mais pour accéder aux manuscrits, il est nécessaire d'avoir un point d'entrée. On présente ici l'autre partie du prototype final, celle qui permet justement l'accès.

Quatre moyens sont disponibles pour trouver le feuillet souhaité, sous la forme de quatre formulaires HTML comme on peut le voir sur la figure 5.25 :

- Navigation dans les **Archives numériques**
- Recherche par **Concept**
- Recherche par **Cote**
- Recherche par **Contenu**

La recherche par concept se fait en choisissant un concept dans une liste. La recherche par cote ou par contenu se fait en saisissant des mots-clefs.

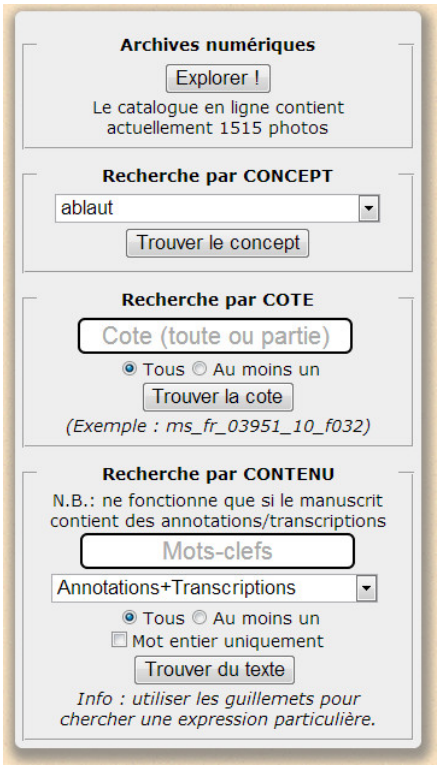


FIGURE 5.25 – *Les quatre formulaires permettant d'accéder aux manuscrits*

5.2.2.1 Recherche

La recherche par **Concept** (le deuxième moyen dans la figure 5.25) permet de trouver tous les manuscrits qui sont liés à un certain concept saussurien. En sélectionnant un concept dans la liste, un résultat de recherche affiche la liste des images auxquelles le concept est lié (voir la figure 5.26).

Les recherches par **Cote** et par **Contenu** (le troisième et le quatrième moyen dans la figure 5.25) fonctionnent toutes les deux sur le même principe. Un champ permet de saisir des mots-clefs (termes de recherche). L'espace sert de délimiteur entre les mots-clefs. Un bouton *radio* permet de choisir si les résultats doivent contenir **Tous** ou **Au moins un** des termes de recherche.

Concept recherché [cliquer pour inverser la coloration] :
signifié
Afficher/Masquer toute la coloration de la recherche

N	Fichier
001	ms_fr_01832_f028.jp2
002	ms_fr_03951_10_f016.jp2
003	ms_fr_03951_10_f033v_034.jp2

FIGURE 5.26 – *Résultat d'une recherche par concept - "signifié" est le concept recherché*

Le fonctionnement du moteur de recherche est détaillé dans la section 6.5.2.

La recherche par **Cote** (le troisième moyen dans la figure 5.25) permet de trouver toutes les images par la cote. Soit la cote complète, soit une ou des portions de la cote peuvent être saisies. Le résultat de recherche affiche la liste des images dont la cote correspond aux critères de recherche (voir la figure 5.27).

Cote recherchée [cliquer pour inverser la coloration] :
3951 22 fr ms
Afficher/Masquer toute la coloration de la recherche

N	Fichier	Cote
001	ms_fr_03951_10_f021v_022.jp2	ms_fr_03951_10_f021v_022
002	ms_fr_03951_10_f022v_023.jp2	ms_fr_03951_10_f022v_023
003	ms_fr_03951_14_f022.jp2	ms_fr_03951_14_f022
004	ms_fr_03951_22_f1.jp2	ms_fr_03951_22_f1
005	ms_fr_03951_7_p22.jp2	ms_fr_03951_7_p22

FIGURE 5.27 – *Résultat d'une recherche par cote avec les termes de recherches :* ***ms fr 3951 22***

La recherche par **Contenu** (le dernier moyen dans la figure 5.25) permet de trouver toutes les images dont les annotations et/ou les transcriptions contiennent les mots-clefs saisis. Le résultat de recherche affiche la liste des images et l'annotation ou la transcription qui correspond aux critères de recherche (voir la figure 5.28).

Terme(s) recherché(s) [cliquer pour inverser la coloration] :
hist anti du de
Afficher/Masquer toute la coloration de la recherche

N	Fichier	Text
001	ms_fr_03951_10_f031v_032.jp2	De L'anti-historicité du langage

FIGURE 5.28 – *Résultat d'une recherche par contenu avec des termes recherches :* ***de anti hist du***

5.2.2.2 Navigation

Dans la figure 5.25, le premier moyen pour trouver un manuscrit est la navigation dans les **Archives numériques**. Il permet de simuler l'ordre d'archivage des manuscrits de la bibliothèque avec les emplacements (section, boîte), puis l'organisation dans la boîte (subdivisions, numéro de feuillet/page), comme déjà expliqué dans la section 4.1.3. Les listes affichées sont dynamiquement extraites du triplestore et ne comportent que les "éléments" qui contiennent au moins une image. Il n'est donc pas possible de naviguer dans une section, ou une boîte, dont aucune image n'existe dans le triplestore.

Le bouton **Explorer** amène au premier niveau connu, c'est-à-dire les **sections** (figure 5.29).

Liste des sections de la BGE (Bibliothèque de Genève) :

N	Section
001	saussure:arch_saussure
002	saussure:ms_fr

FIGURE 5.29 – *Page de navigation : liste des sections de la BGE*

En choisissant la **section**, on peut voir toutes les boîtes qu'elle contient (figure 5.30). Ici, il est possible de choisir soit la **boîte**, soit **tous les feuillets de la boîte (toutes subdivisions confondues)** (explication ci-dessous).

Liste des boîtes de la section : http://saussure.com/ressource/arch_saussure

N	Boîte	Tous les feuillets de la boîte (toutes subdivisons confondues)
001	saussure:arch_saussure_361	saussure:arch_saussure_361
002	saussure:arch_saussure_366	saussure:arch_saussure_366
003	saussure:arch_saussure_367	saussure:arch_saussure_367
004	saussure:arch_saussure_368	saussure:arch_saussure_368

FIGURE 5.30 – *Page de navigation : liste des boîtes de la section Arch. Saussure*

En choisissant la **boîte**, on peut voir toutes les **subdivisions** qu'elle contient (figure 5.31). Lorsque les feuillets ne sont rattachés à aucune subdivision (soit ils sont directement dans la boîte, soit le numéro de l'enveloppe qui les contient ne fait pas partie de la cote), alors on utilise une "fausse" subdivision "**sans subdivision**" (figure 5.32).

Liste de subdivisions pour la boîte : http://saussure.com/ressource/arch_saussure_368

N	Subdivision
001	saussure:arch_saussure_368_4
002	saussure:arch_saussure_368_5

FIGURE 5.31 – *Page de navigation : liste des subdivisions de la boîte 368*

Liste de subdivisions pour la boîte : http://saussure.com/ressource/arch_saussure_398

N	Subdivision
001	{Sans subdivision} tous les feuillets rattachés à la boîte, mais à aucune subdivision.

FIGURE 5.32 – *Page de navigation : cas d'une "fausse" subdivision lorsqu'elle n'est pas indiquée*

En choisissant une **subdivision**, on peut voir toutes les **images (fichiers)** qu'elle contient (figure 5.33).

Depuis la section, on pouvait choisir soit la **boîte**, soit **tous les feuillets de la boîte (toutes subdivisions confondues)**. Cette deuxième option est utile lorsque la boîte contient beaucoup de subdivisions qui ne comportent qu'un ou deux feuillets. Cependant, lors de l'évaluation de l'interface (chapitre 7), les utilisateurs ont eu de la peine à comprendre l'intérêt

Liste de feuillets de la subdivision : http://saussure.com/ressource/arch_saussure_368_4

N	Fichier
001	arch_saussure_368_4_f41_42v.jp2
002	arch_saussure_368_4_f41v_42.jp2
003	arch_saussure_368_4_f43.jp2
004	arch_saussure_368_4_f43v_44.jp2
005	arch_saussure_368_4_f44v.jp2
006	arch_saussure_368_4_f45.jp2
007	arch_saussure_368_4_f45v.jp2

FIGURE 5.33 – *Page de navigation : liste des feuillets de la subdivision 4 de la boîte 368*

de cette option et elle sera sûrement supprimée. Si on la choisit, on peut voir tous les feuillets contenus dans la boîte (qu'ils soient dans une subdivision ou non, et quelle que soit leur subdivision) (figure 5.34).

Liste de tous les feuillets (toutes subdivisions confondues) de la boîte : http://saussure.com/ressource/arch_saussure_368

N	subdivision	Fichier
001	**saussure:**arch_saussure_368_4	arch_saussure_368_4_f41_42v.jp2
002	**saussure:**arch_saussure_368_4	arch_saussure_368_4_f41v_42.jp2
003	**saussure:**arch_saussure_368_4	arch_saussure_368_4_f43.jp2
004	**saussure:**arch_saussure_368_4	arch_saussure_368_4_f43v_44.jp2
005	**saussure:**arch_saussure_368_4	arch_saussure_368_4_f44v.jp2
006	**saussure:**arch_saussure_368_4	arch_saussure_368_4_f45.jp2
007	**saussure:**arch_saussure_368_4	arch_saussure_368_4_f45v.jp2
008	**saussure:**arch_saussure_368_5	arch_saussure_368_5_f68.jp2
009	**saussure:**arch_saussure_368_5	arch_saussure_368_5_f68v.jp2

FIGURE 5.34 – *Page de navigation : liste de tous les feuillets de la boîte 368 (toutes subdivisions confondues*

Chapitre 6

Implémentation

L'analyse des besoins des utilisateurs, ainsi que les recherches effectuées pour l'état de l'art ont permis l'élaboration d'un prototype sous la forme d'une application web.

Le développement réalisé consiste en une application client (avec une interface graphique) pour les utilisateurs (HTML, Javascript), communiquant avec un service frontal (PHP) sur un serveur web. L'affichage de l'image dans l'interface graphique se fait grâce au serveur IIPImage et toutes les métadonnées (y compris les annotations et les transcriptions) sont stockées dans un serveur OpenRDF Sesame (triplestore) auquel on accède par un *endpoint* SPARQL.

Ce chapitre présente l'architecture globale de l'application, puis décrit l'implémentation en commençant par le triplestore, puis les réalisations du côté du serveur, pour arriver à l'application client, avant de terminer avec un exemple détaillé de l'une des fonctionnalités centrales.

Comme il s'agit d'une section technique, et que le développement et la mise en place (système) sont conséquents, on part du principe que le lecteur a des connaissances en algorithmique, ainsi que dans les technologies ou domaines suivants :

- les langages HTML, Javascript, PHP, Bash et Python
- le fonctionnement d'applications client/serveur et d'API
- le protocole HTTP (GET/POST, réponses, etc. notamment avec la librairie cURL)
- les problématiques d'encodage de l'information (ISO-8859-1, UTF-8, Unicode, URL encoded, entités XML/HTML, etc.)
- les formats d'échanges XML et JSON
- les expressions régulières (regex) standards et Unicode
- RDF et le langage de requête SPARQL

Le code source commenté est fourni avec ce mémoire. Il n'est donc pas détaillé ici, sauf pour les parties qui le nécessitent, sous la forme d'extraits ou de pseudo-code.

6.1 Architecture et déploiement

Cette section décrit l'architecture globale de l'application (prototype) réalisée dans ce travail. En effet, une vue d'ensemble permet une meilleure compréhension des interactions entre les différentes parties.

6.1.1 Architecture de l'application globale

La figure 6.1 schématise l'architecture du système et les rôles principaux de l'application, alors que le schéma 6.2 synthétise les rôles principaux de l'application. Un navigateur web communique avec deux services distants : le serveur web (PHP) et le serveur d'images IIPImage. Toutes les données sont stockées dans un troisième service : le triplestore (OpenRDF Sesame). N.B. : Dans la réalité, ces trois services sont installés sur la même machine, même si les schémas montrent des serveurs séparés pour clarifier les explications.

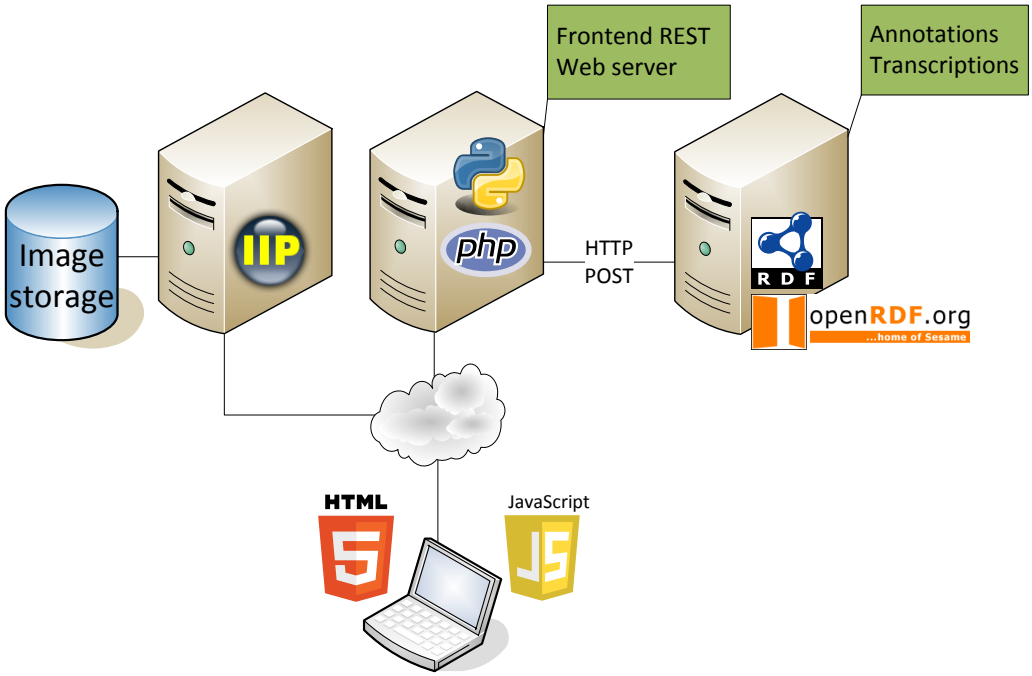


FIGURE 6.1 – *Architecture système / schéma des technologies*

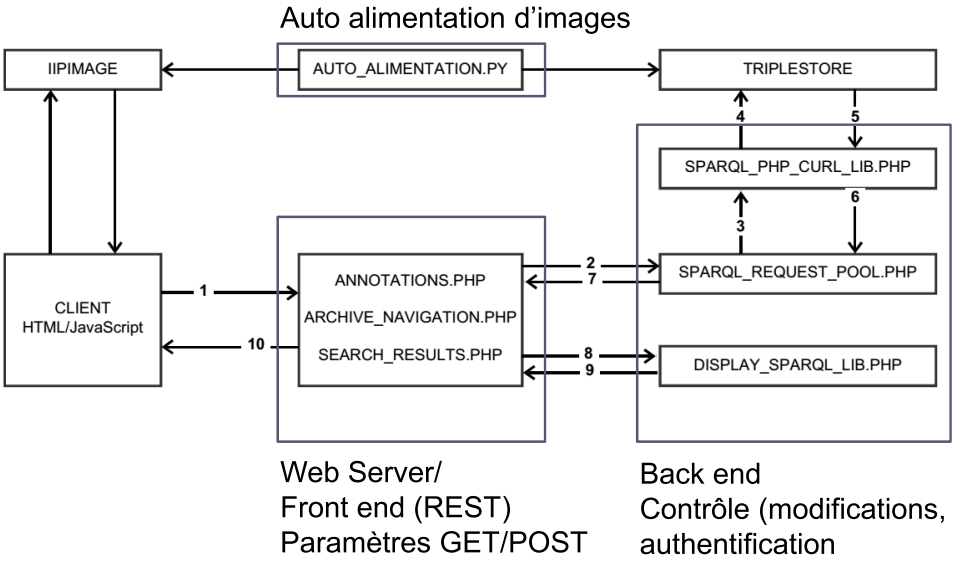


FIGURE 6.2 – *Schéma des rôles de l'application et des échanges (communications)*

L'authentification des utilisateurs du site est gérée automatiquement par UserCake. N'importe qui peut s'inscrire et créer un compte gratuitement. Pour limiter le *spam*, les scripts de UserCake ont été modifiés afin d'améliorer la sécurité de base. La création d'un compte nécessite une adresse email (envoi d'un lien de confirmation d'inscription) et la saisie d'un captcha. Le système de captcha par défaut a été remplacé par celui de reCAPTCHA [1], qui est plus sûr.

Tous les échanges entre le client et le serveur public se font par le protocole HTTP. L'échange d'information pour IIPImage se fait avec le protocole IIP, et via des paramètres GET/POST pour le service frontal (PHP).

Le serveur IIPImage accède à un dépôt d'images (JPEG2000), ce qui lui permet de servir les *tiles* au client.

Le service frontal (PHP) agit comme intermédiaire entre l'application client et le triplestore. Il reçoit des requêtes des clients, analyse leur contenu, puis le transfère au *backend*. Ce dernier le transforme en une requête SPARQL, qu'il transmettra au triplestore. La réponse à la requête sera mise en forme avant d'être renvoyée au client. Les requêtes SPARQL permettent de récupérer les données liées à l'image affichée, mais aussi d'en ajouter, d'en modifier ou d'en supprimer.

Les données sont stockées dans un triplestore OpenRDF Sesame et accessibles via le *endpoint* SPARQL. Les accès au *endpoint* en consultation (GET) sont publics (aucune restriction), et les accès en écriture (POST/PUT/DELETE) nécessitent une authentification gérée au niveau du service frontal. L'application client passe par le service frontal pour toutes les requêtes pour simplifier le traitement et la mise en forme. De plus, les modifications effectuées via le *endpoint* sont contrôlées par le service frontal et seuls les utilisateurs authentifiés peuvent faire des modifications depuis l'interface utilisateur.

Un script Python sur le serveur sert à l'auto-alimentation du contenu : des images peuvent être déposées sur le serveur (FTP, disque dur, etc.), et le script s'occupe de les convertir en JPEG2000, de les placer dans le dépôt et d'ajouter les informations nécessaires dans le triplestore.

6.1.2 *Workflow* du point de vue du client

Le client peut afficher dans son navigateur les pages suivantes :
- homepage.php
- viewer_visitor.php
- search_results.php
- archive_navigation.php

La page d'accueil du site (**homepage.php**) présente les quatre formulaires décrits dans la section 5.2.2. Le premier formulaire (navigation) affiche la page **archive_navigation.php**. Les trois autres (recherche par concept, par cote et par contenu) affichent la page **search_results.php**. Ces formulaires permettent tous de trouver les manuscrits en ligne et de les afficher avec la page **viewer_visitor.php**. Celle-ci contient l'application client (Javascript) qui communique avec le service frontal **annotations.php**.

Les scripts **archive_navigation.php**, **search_results.php** et **annotations.php** fonctionnent tous les trois sur le même principe : le client envoie des requêtes HTTP au script et les paramètres (GET/POST) sont analysés, traités, puis une réponse est retournée. Il existe

1. `http://www.google.com/recaptcha`

cependant une différence : le script **annotations.php** est un service qui retourne une information semi-structurée au client, tandis que **archive_navigation.php** et **search_results.php** retournent une page complète. Toutes ces étapes sont décrites dans le reste de ce chapitre.

6.1.3 Déploiement des systèmes

Comme on l'a vu dans la section 6.1.1, plusieurs serveurs ou services sont nécessaires avant de pouvoir développer l'application décrite ci-après. Une part importante du travail a été de mettre en place les différents serveurs. Bien qu'implicitement indispensable, cette étape n'est pas clairement définie dans le cahier des charges.

Les installations de chaque serveur, les dépendances logicielles, ainsi que leurs paramétrages ne sont pas détaillés ici. Plusieurs scripts sont fournis en annexe pour l'installation et le déploiement automatique de l'intégralité de l'environnement de travail.

Deux procédures d'installation sont disponibles en annexe (Annexe B). Ces procédures et les explications de cette section sont suffisantes à la compréhension de ces installations. La lecture des scripts commentés permettra de répondre aux questions techniques.

La première procédure (Annexe B.2.2) utilise un script *preseed* Debian pour l'installation automatique (*unattended*) du système d'exploitation. Celui-ci formate **automatiquement** les disques durs de la machine, télécharge depuis internet les paquets nécessaires, puis procède à l'installation, et crée un utilisateur. Un script Bash est automatiquement exécuté avant la toute dernière étape de l'installation. Cette technique permet de modifier ou de créer des fichiers de configuration (c'est d'ailleurs ici qu'est généré le script d'installation décrit ci-dessous).

La deuxième procédure (Annexe B.2.3) utilise un script Bash à exécuter sur la machine fraîchement installée. Celui-ci installe et configure automatiquement le serveur IIPImage, OpenRDF Sesame et leurs dépendances (Apache, Tomcat, etc.). Un site de démonstration est également déployé. Il est seulement nécessaire de convertir quelques images (fournies) avec le script d'auto-alimentation pour avoir un site fonctionnel. Plusieurs questions sont posées à l'utilisateur durant l'exécution du script (adresse IP du site, adresse email, etc.). Les informations saisies sont nécessaires aux configurations de la base de données MySQL (pour l'authentification avec UserCake). Une adresse email (et son mot de passe) est nécessaire pour l'envoi du mail de confirmation lors de l'inscription d'un utilisateur (validation d'inscription avec UserCake).

6.2 Authentification et sécurisation du serveur

Pour limiter les abus, les requêtes de modifications des données du triplestore passent toutes par le service frontal (PHP). Celui-ci n'autorise les modifications que si les utilisateurs sont authentifiés (voir l'authentification avec UserCake ci-dessous). Le service frontal communique avec le script sparql_lib_pool.php, qui possède dans ses sources les informations nécessaires pour s'authentifier sur l'application Tomcat OpenRDF Sesame.

Comme l'on souhaite contrôler les modifications, un seul utilisateur a les droits en édition sur le triplestore. De cette façon, seul un utilisateur connecté localement sur le serveur, ainsi que les scripts d'auto_alimentation_4.py et sparql_lib_pool.php peuvent apporter des modifications aux données du triplestore. Cela évite que quelqu'un de mal intentionné, même authentifié, ne puisse écrire des requêtes malicieuses et les envoyer par cURL.

6.2.1 Authentification avec UserCake

UserCake est un système de gestion des utilisateurs développé en PHP. Il simplifie la mise en place de l'authentification sur un site web. UserCake stocke les informations des utilisateurs (nom, adresse mail, mot de passe, etc.) dans une base de données relationnelle (MySQL) et la vérification de l'authentification se fait grâce aux **sessions PHP**.

Comme l'authentification ne fait pas partie du cahier des charges, on renvoie le lecteur intéressé à la documentation de UserCake et au code source de ce projet.

6.2.2 Sécurisation du triplestore OpenRDF Sesame (Tomcat)

Malheureusement, l'authentification n'est pas intégrée dans OpenRDF Sesame. Il faut donc contourner ce problème pour sécuriser les accès au *endpoint* SPARQL, afin de garder le contrôle sur les modifications des données. Pour ce faire, on s'inspire largement d'un article publié sur internet [2] qui décrit la sécurisation de l'application Tomcat.

On définit deux rôles :

- **sesame-admin** Ce rôle possède les droits d'écriture (POST/PUT/DELETE) sur /repositories/SYSTEM/*, c'est-à-dire la configuration du serveur. Il peut notamment créer des *repositories*, ou les supprimer entièrement.
- **editor** Ce rôle possède les droits d'écriture (POST/PUT/DELETE) sur /repositories/mass4/statements qui contient les données du site des archives numériques.

La déclaration de ces rôles se trouve dans le fichier **/var/lib/tomcat6/webapps/openrdf-sesame/WEB-INF/web.xml** (voir les listings 6.1 et 6.2). Aucune restriction n'a été imposée pour la méthode GET, ce qui signifie que tout le monde peut lire les données.

```
<security-constraint>
   <web-resource-collection>
      <web-resource-name>config</web-resource-name>
      <url-pattern>/repositories/SYSTEM</url-pattern>
      <url-pattern>/repositories/SYSTEM/*</url-pattern>
      <http-method>POST</http-method>
      <http-method>PUT</http-method>
      <http-method>DELETE</http-method>
   </web-resource-collection>
   <auth-constraint>
      <role-name>sesame-admin</role-name>
   </auth-constraint>
</security-constraint>
```

Listing 6.1 – Tomcat : extrait de **web.xml** pour openrdf-sesame (1)

```
<security-constraint>
   <web-resource-collection>
      <web-resource-name>editor</web-resource-name>
      <url-pattern>/repositories/mass4/statements</url-pattern>
      <http-method>POST</http-method>
      <http-method>PUT</http-method>
      <http-method>DELETE</http-method>
   </web-resource-collection>
   <auth-constraint>
      <role-name>editor</role-name>
   </auth-constraint>
</security-constraint>
```

Listing 6.2 – Tomcat : extrait de **web.xml** pour openrdf-sesame (2)

2. http://rivuli-development.com/further-reading/sesame-cookbook/basic-security-with-http-authentication/

Pour compléter le rôle **sesame-admin**, on modifie aussi les configurations de openrdf-workbench (l'interface web pour explorer/modifier les données sur Sesame) dans le fichier **/var/lib/tomcat6/webapps/openrdf-workbench/WEB-INF/web.xml** (listing 6.3).

```
<security-constraint>
   <web-resource-collection>
      <web-resource-name>Sesame Workbench</web-resource-name>
      <url-pattern>/*</url-pattern>
   </web-resource-collection>
   <auth-constraint>
      <role-name>sesame-admin</role-name>
   </auth-constraint>
</security-constraint>
```

Listing 6.3 – Tomcat : extrait de **web.xml** pour openrdf-workbench

Finalement, on attribue ces rôles à des utilisateurs. On définit deux utilisateurs et leur mot de passe dans **/etc/tomcat6/tomcat-users.xml** (voir le listing 6.4).

```
<user username="sesameEditor" password="randomPassword" roles="editor" />
<user username="tomcatAdmin" password="randomPassword2" roles="sesame-admin" />
```

Listing 6.4 – Tomcat : extrait de **tomcat-users.xml**

De cette manière, toute modification des données du triplestore nécessitent une authentification avec un nom d'utilisateur et un mot de passe.

6.3 Auto alimentation du système (Python)

6.3.1 Corpus d'images : problème de conversion et images de mauvaise qualité

Après moult discussions avec les utilisateurs du projet au sujet des cotes et de l'identification des manuscrits (voir section 4.1.3), un script Bash a été développé pour la détection de la cote depuis le nom de fichier. Il s'est avéré que les explications fournies par les utilisateurs ne couvraient qu'un sous-ensemble des cas. En effet, à la réception du corpus de test, seulement 55% des 2231 images possédaient un nom de fichier correspondant à la convention de nommage.

Une nouvelle convention de nommage a dû être définie et une refonte complète du script Bash a dû être réalisée. En effet, les nouvelles indications n'ont pas pu être appliquées dans le script existant à cause de certaines limitations du langage. Il a donc été intégralement réécrit en Python, car ce langage offre plus de possibilités.

En analysant en détail les noms des fichiers problématiques, il a été possible de réécrire l'algorithme afin de couvrir beaucoup plus de cas. Avec l'aval des utilisateurs du projet, des images ont été exclues du corpus, car elles étaient de trop mauvaise qualité (par ex : des scans de microfilms, etc.), et certaines images ont été renommées car les noms de fichiers présentaient des erreurs évidentes (erreurs de saisie, caractères "interdits", etc.). La liste des images exclues du lot initial, ainsi que la liste des images renommées du lot initial sont respectivement disponibles dans les fichiers fournis en annexe **liste_images_exclues_(409).txt** et **liste_fichiers_renommés_v0.txt**.

Après ce premier tri, il restait 1822 images (**liste_toutes_images_v0_(1822).txt**) parmi lesquelles se trouvaient encore une quinzaine de fichiers avec des noms identiques (**liste_noms_doublons_(15_entre_1822_1807).txt**) qui ont été ignorés.

Après ce second tri, il restait un corpus final de 1807 images qui ont toutes été traitées et sur lesquelles 1729 cotes ont été correctement détectées. Les 78 fichiers restants se composaient de 40 fichiers dont les noms étaient ambigus, 33 fichiers dont les noms n'étaient pas acceptables, et 5 fichiers dont la conversion en JPG2000 avec kdu_compress a retourné une erreur. La liste complète des 1807 images se trouve dans le fichier annexe : **liste_toutes_images_v1_(1807).txt**.

Finalement, les 1729 images ont été mises en ligne, soit 95.68% du corpus. Les 78 images problématiques ont été mises de côté et devront être analysées de plus près (voir l'Annexe B.6.1). Après la mise en ligne, 214 images problématiques ont été détectées. Leur visualisation ne se faisait pas correctement et, pire, certaines bloquaient le serveur IIPImage pendant plusieurs minutes, rendant le site inutilisable. Ces images ont été retirées du site (et du triplestore) pour laisser un total final de 1515 images en ligne. La liste des 214 images problématiques et la liste des 1515 images en ligne se trouvent respectivement dans les fichiers annexes : **liste_images_conversion_problématique_(214).txt** et **liste_toutes_images_v2_(1515).txt**.

6.3.2 Fonctionnement du script Python

Le script Python **auto_alimentation_4.py** ne prend pas de paramètres. Cependant, des variables sont à modifier au début du script en fonction des configurations de la machine sur laquelle il est exécuté (dossier source avec les images, dossier destination, chemin de kdu_compress... voir le listing 6.5).

Les accès au triplestore se font avec la librairie **pycurl**, et utilisent l'authentification décrite précédemment.

Un système de *log* (journal) enregistre minutieusement toutes les opérations pour pouvoir remonter les éventuels problèmes. Un extrait du fichier log est disponible en annexe (Annexe B.5.1).

```
############################################################
# SET THESE VARIABLES
############################################################
FTP_DEPOSIT="/var/www/massimo-dev/www/auto_alim/lot_test/"
PUBLIC_STORAGE="/var/www/massimo-dev/www/auto_alim/public_storage/"
ARCHIVE_ORIGINALS="/var/www/massimo-dev/www/auto_alim/archive_originals/"
working_dir="/var/www/massimo-dev/www/auto_alim/processing/"
ERROR_DIR="/var/www/massimo-dev/www/auto_alim/error/"
kdu_compress="/root/kakadu/Linux-x86-32/kdu_compress"

SESAME_USER_AND_PASSWORD="sesameEditor:randomPassword"

prefixes='''
PREFIX rdfs:<http://www.w3.org/2000/01/rdf-schema#>
PREFIX p:<http://saussure.com/property/>
PREFIX saussure:<http://saussure.com/ressource/>
PREFIX xsd:<http://www.w3.org/2001/XMLSchema#>
PREFIX rdf:<http://www.w3.org/1999/02/22-rdf-syntax-ns#>
'''

graph_file="file://test_001_2013-04-08.xml"
endpoint_update="http://192.168.56.100:8080/openrdf-sesame/repositories/mass4/statements"
endpoint_query="http://192.168.56.100:8080/openrdf-sesame/repositories/mass4"
```

Listing 6.5 – Variables à modifier dans le script **auto_alimentation_4.py**

Le script lit le contenu du dossier source contenant les images, puis traite chaque fichier l'un après l'autre. Pour chaque image, il essaie d'extraire la cote depuis le nom de fichier avec une regex (voir le listing 6.6). Si la cote correspond à un *pattern* accepté, elle est découpée

automatiquement en unités (section, boîte, subdivision, feuillet). Ces unités permettent la construction des URI pour identifier chaque unité (voir la génération d'URI uniques dans la section 4.1.3).

La présence dans le triplestore de certaines URI est vérifiée avec des requêtes SPARQL ASK (voir les listings 6.7 et 6.8). Si les résultats des requêtes ASK sont tous négatifs, alors on considère que l'image n'existe pas déjà dans le triplestore. Dans ce cas, la requête SPARQL pour l'ajout est sauvée temporairement dans une variable (listing 6.9).

```
regex  = "^(ms_fr|arch_saussure)"
regex += "(_[0-9]+(bis)?)"
regex += "([_a-zA-Z0-9]+)?"
regex += "("
regex +=   "(_(0|z)+)?"
regex +=   "("
regex +=      "(_env)|"
regex +=      "(_dossier)|"
regex +=      "(_garde_(ant|post)(_v)?)|"
regex +=      "(_1ere_couv)|(_[2-4]eme_couv)|"
regex +=      "(_(f|p|piece_)[0-9]+(v|r|bis|bis_v)?(_[0-9]+(v|bis|bis_v)?)?)"
regex +=   "){1,2}"
regex += ")"
regex += "(_[_a-zA-Z0-9]+)?$"
```

Listing 6.6 – Auto-alimentation : regex du script Python

L'image source est convertie en JPEG2000 avec un programme externe (kdu_compress de la libraire Kakadu). Si la conversion s'est bien déroulée, alors tous les éléments sont prêts pour l'ajout : l'image convertie est placée dans le dépôt d'images (pour IIPImage), l'image originale est conservée dans un dossier séparé et la requête SPARQL pour l'ajout est exécutée (listing 6.9). Si ces opérations sont effectuées avec succès, alors l'image est disponible en ligne. En cas d'échec de l'une des opérations ci-dessus, le traitement pour l'image courante est abandonné, et le traitement recommence avec l'image suivante.

```
n3_triple="saussure:%s rdf:type saussure:Surface_d_ecriture ." % surfaceEcriture
n3_triple='''?anything p:hasCote "%s" .''' % surfaceEcriture
n3_triple='''saussure:%s rdf:type saussure:Photo .''' % photo
n3_triple='''?anything p:hasFilename "%s" .''' % jp2Filename
```

Listing 6.7 – Requête SPARQL ASK - construction du contenu de la requête

```
'''%s
ASK { GRAPH <%s>
        {
        %s
        }
} '''   % (prefixes, graph_file, n3_triple)
```

Listing 6.8 – Requête SPARQL ASK - requête ASK générique

```
saussure:ms_fr rdf:type saussure:Section ;
        p:hasArchiveBox saussure:ms_fr_03951 .
saussure:ms_fr_03951 rdf:type saussure:ArchiveBox ;
        p:hasSubdivisions saussure:ms_fr_03951_10 .
saussure:ms_fr_03951_10 rdf:type saussure:Subdivisions ;
        p:hasSurfaceEcriture saussure:ms_fr_03951_10_f028v_029 .
saussure:ms_fr_03951_10_f028v_029 rdf:type saussure:Surface_d_ecriture ;
        p:hasCote "ms_fr_03951_10_f028v_029";
        p:hasPhoto saussure:ms_fr_03951_10_f028v_029-DOT-jp2 .
saussure:ms_fr_03951_10_f028v_029-DOT-jp2 rdf:type saussure:Photo ;
        p:hasCote "ms_fr_03951_10_f028v_029" ;
        p:hasSource "http://saussure.unige.ch/bge/manuscript/ms_fr_03951_10_f028v_029.jp2" ;
        p:hasFilename "ms_fr_03951_10_f028v_029.jp2" .
```

Listing 6.9 – Requête SPARQL d'ajout de l'image ms_fr_03951_10_f028v_029

6.3.3 Convention de nommage

Pour que l'automatisation fonctionne, les fichiers doivent être nommés selon une convention. Celle-ci a été convenue avec les utilisateurs du projet et cette section se propose de l'illustrer par des exemples.

La regex (voir listing 6.6) est utilisée pour détecter la cote depuis le nom de fichier sans son extension. Les fichiers ne peuvent contenir que des caractères alphanumériques, des *underscores* et des tirets. Tous les séparateurs (espace, point, virgule, *slash*, etc.) doivent être remplacés par des *underscores*. Notons que cette détection n'est valable que pour les manuscrits de Ferdinand de Saussure de la Bibliothèque de Genève. Les autres bibliothèques ont aussi un système de cote, mais qui leur est propre.

Le découpage de la regex du listing 6.6 fonctionne de cette façon :

- **Ligne 1** Section
- **Ligne 2** Boîte
- **Ligne 3** [optionnel] Subdivision(s)
- **Ligne 4-13** Feuillet, page, dossier, enveloppe... (1 ou 2 fois)
  - **Ligne 5** [optionnel] Préfixe pour forcer un ordre (0000 pour le début, zzzz pour la fin)
  - **Ligne 6-12** Feuillet, page, dossier, enveloppe...
    - ◇ **Ligne 7** enveloppe, ou
    - ◇ **Ligne 8** dossier, ou
    - ◇ **Ligne 9** garde antérieure ou garde extérieure et éventuellement leurs versos, ou
    - ◇ **Ligne 10** 1ère, 2ème, 3ème ou 4ème de couverture, ou
    - ◇ **Ligne 11** numéro de feuillet/page/pièce, et éventuellement l'indication du "verso"/"recto"/"bis verso"/"bis". [optionnel] Un deuxième numéro de feuillet/page/pièce, éventuellement suivi de l'indication du "verso"/"bis verso"/"bis"
- **Ligne 14** [optionnel] Suffixe libre ("ameliore", "test", "charte_regle", "greyscale"...)

Ainsi, une cote doit contenir au minimum une section, une boîte et un feuillet (qui respectent les *patterns*). Si ces trois éléments ne sont pas détectés, la cote est refusée. S'ils sont détectés, mais qu'il y a un "reste" (c'est-à-dire le suffixe de la ligne 14), alors la cote est considérée comme ambiguë et elle est refusée.

Pour faciliter la gestion des cas problématiques (cotes refusées ou ambiguës), un formulaire HTML a été développé (annexe B.6.1) permettant de renommer les fichiers, de forcer manuellement le découpage de la cote ou de supprimer un fichier. Le renommage, l'effacement et le découpage manuel de la cote sont fonctionnels, mais il reste encore à intégrer le formulaire dans le *workflow* de l'application pour que les fichiers traités soient pris en compte (déplacer les fichiers, exécuter le script **auto_alimentation_4.py**, etc.).

6.3.4 Quelques cas d'extraction de cotes détaillés

Cette section détaille quelques cas représentatifs, ou au contraire particuliers, pour illustrer le fonctionnement du script Python et le découpage de la cote selon la convention de nommage.

6.3.4.1 Exemples de cotes acceptées

- **arch_saussure_398_f020**
  - Section : arch_saussure
  - Boîte : 398
  - Subdivision : /
  - Feuillet : f020

- **ms_fr_03951_10_f031v_032**
  - Section : ms_fr
  - Boîte : 03951
  - Subdivision : 10
  - Feuillet : f031v_032
- **ms_fr_03952_4_b_0000_1ere_couv_4eme_couv**
  - Section : ms_fr
  - Boîte : 03952
  - Subdivision : 4_b
  - Feuillet : 0000_1ere_couv_4eme_couv
- **ms_fr_03951_1_1er_cours_p17**
  - Section : ms_fr
  - Boîte : 03951
  - Subdivision : 1_1er_cours
  - Feuillet : p17

Ces cotes sont représentatives des cotes acceptées. La section, la boîte, les subdivisions (s'il y en a) et les feuillets/pages sont clairement définis.

Le premier cas est un feuillet qui se situe directement dans la boîte (sans subdivision).

Le deuxième cas est un cahier ouvert dont les pages se trouvent dans la subdivision 10.

Le troisième cas est moins courant. Il y a une subdivision "composée" (deux niveaux d'enveloppes : 4 et b). Certaines photos ont été nommées en préfixant le feuillet d'un "code", de façon à forcer un ordre (si elles sont triées alphabétiquement). Deux groupes sont acceptés "0000" et "zzzz". Ils permettent de faire apparaître la couverture, l'enveloppe, etc. respectivement au début ou à la fin du lot.

Le quatrième cas possède une subdivision peu courante, mais acceptée, car le nommage de la page (p17) est bien identifiable.

6.3.4.2 Exemples de cotes ambiguës

- **ms_fr_03963_1_f019v_ameliore**
  - Section : ms_fr
  - Boîte : 03963
  - Subdivision : 1
  - Feuillet : f019v
  - Reste : ameliore
- **ms_fr_03965_12_zzzz_4eme_couv_charte_regle**
  - Section : ms_fr
  - Boîte : 03965
  - Subdivision : 12
  - Feuillet : zzzz_4eme_couv
  - Reste : charte_regle

Le premier cas est une version "améliorée" de l'image, c'est-à-dire que le photographe a appliqué un traitement (contraste, etc.).

Le deuxième est une photo de la quatrième de couverture (le "zzzz" pour le placer en fin de lot). Cette photo comprend également une règle métrique posée à côté du document pour les dimensions, et la charte de couleurs utilisée par le photographe professionnel pour avoir une référence de couleur neutre.

6.3.4.3 Exemples de cotes refusées

- **ms__fr__05133__0001** - Aucun feuillet détecté
  - Section : ms_fr
  - Boîte : 05133
  - Subdivision : 0001
- **arch__saussure__tableau** - Aucune boîte détectée
  - Section : arch_saussure
- **ms__fr__03972__15__p308b__c** - Aucun feuillet détecté
  - Section : ms_fr
  - Boîte : 03972
  - Subdivision : 15_p308b_c
- **ms__fr__03951__10__couverture** - Aucun feuillet détecté
  - Section : ms_fr
  - Boîte : 03951
  - Subdivision : 10_couverture

On constate que la détection des *tokens* est fausse, raison pour laquelle ces images sont refusées lors du traitement automatique.

Dans le premier cas, la section et la boîte sont bien détectées. Le dernier *token* 0001 ne peut pas être détecté comme un feuillet, il est donc détecté comme une subdivision. Solution : ajouter le feuillet dans le nom de fichier (par ex. : **ms__fr__05133__f001**).

Le deuxième cas est vraiment un cas particulier, car la section existe, mais n'y a pas de boîte. Solution : ajouter la boîte dans le nom de fichier (par ex. : **arch__saussure__03951__tableau**).

Le troisième cas est correctement nommé par rapport à la cote réelle, mais il contient des éléments trop peu courants (p308b_c) pour faire partie de la regex. Solution : forcer le découpage des *tokens* dans le formulaire prévu à cet effet.

Le quatrième cas est correct, mais ne respecte pas la convention adoptée. Solution : s'il s'agit de la couverture d'un cahier ou d'un registre, il devrait spécifier qu'il s'agit, par exemple, de la première de couverture (par ex. : **ms__fr__03951__10__0000__1ere__couv**).

6.4 Backend (PHP)

Le *backend* se situe entre le triplestore et les scripts/pages utilisés par le client. Tous les contrôles (authentification) et toutes les vérifications (validité des données) sont effectués ici. Les trois fichiers **sparql__php__curl__lib.php**, **sparql__request__pool.php** et **display__sparql__lib.php** forment le *backend* du système. Il s'agit d'un ensemble de fonctions (envoi de requêtes, construction de requêtes et mise en forme des résultats). Elles sont appelées par le *frontend* (annotations.php), ainsi que pour la recherche et la navigation (search_results.php et archive_navigation.php). La figure 6.3 schématise quels fichiers effectuent les appels au *backend*, alors que la figure 6.4 schématise l'organisation des échanges de l'ensemble de l'application.

Cette section décrit les fonctions contenues dans les trois fichiers du *backend* en expliquant brièvement leur fonctionnement. Le code source fourni en annexe répondra aux interrogations techniques des plus curieux.

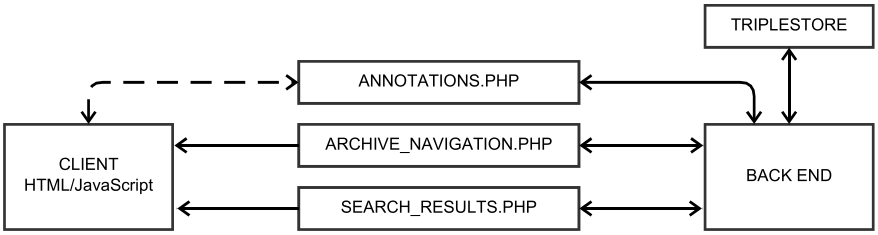


FIGURE 6.3 – *Schéma des appels au backend*

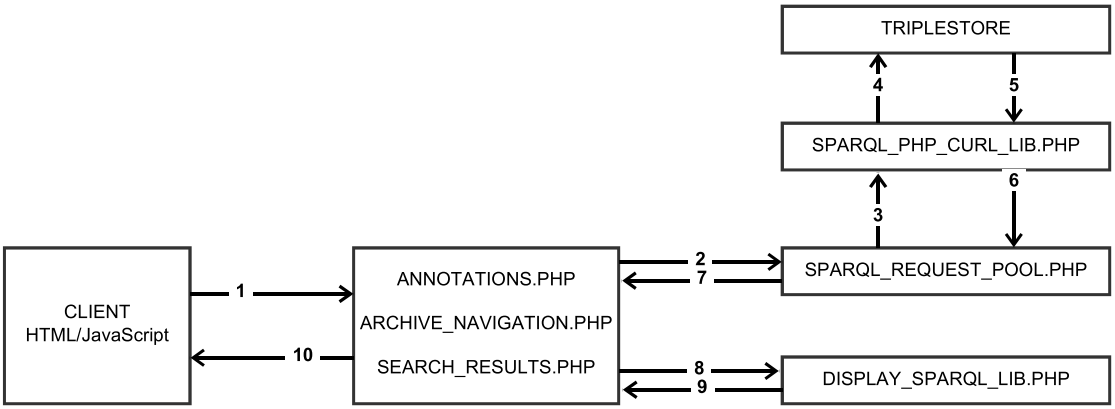


FIGURE 6.4 – *Échanges (appels de fonctions ou communications) entre les différentes parties du système*

6.4.1 Accès au *endpoint* SPARQL (librairie/cURL)

Il existe quelques librairies pour l'interaction avec Sesame (voir Annexe C.1.1). Ces librairies ne font qu'ajouter une couche finalement inutile entre le triplestore et notre application. La solution adoptée a finalement été d'utiliser la librairie PHP cURL (Client URL) [3] et de communiquer directement avec l'API HTTP de Sesame (*endpoint* SPARQL).

Le *endpoint* SPARQL permet différentes opérations selon la méthode utilisée :

> *"GET : Fetches statements from the repository.*
> *PUT : Updates data in the repository, replacing any existing data with the supplied data.*
> *DELETE : Deletes statements from the repository.*
> *POST : Performs updates on the data in the repository."* [4]

Il est ainsi possible d'effectuer toutes les tâches de recherche avec GET, et toutes les tâches d'édition avec POST ; la méthode POST accepte aussi les requêtes de recherche. Les méthodes PUT et DELETE ne sont pas utilisées dans ce travail.

En se basant sur l'article d'un blog [5], sur la documentation de Sesame (Chapter 8. HTTP communication protocol for Sesame 2 [6]) et sur les recommandations du W3C (SPARQL 1.1 Graph Store HTTP Protocol [7] et SPARQL 1.1 Update [8]), il a été possible d'écrire une "librairie" adaptée au projet pour simplifier l'interaction entre l'application web (PHP) et Sesame : **sparql_php_curl_lib.php**. La description des fonctions de **sparql_php_curl_lib.php** se trouve dans la documentation technique en annexe (voir Annexe D.1.1).

3. http://php.net/manual/en/book.curl.php
4. http://www.openrdf.org/doc/sesame2/system/ch08.html
5. http://johnwright.me/blog/sparql-query-in-code-rest-php-and-json-tutorial/
6. http://www.openrdf.org/doc/sesame2/system/ch08.html
7. http://www.w3.org/TR/sparql11-http-rdf-update/
8. http://www.w3.org/TR/sparql11-update

6.4.2 Construction et exécution des requêtes SPARQL

Le fichier **sparql_request_pool.php** contient toutes les fonctions PHP qui permettent de faire des requêtes SPARQL. Sur la base de paramètres, chaque fonction génère la requête SPARQL, puis l'exécute avec les fonctions de **sparql_php_curl_lib.php** (voir Annexe D.1.1), puis retourne le résultat.

Lors de l'ajout, il est nécessaire d'insérer l'information avec des identifiants uniques (URI). Ceux-ci sont générés dans la procédure d'ajout et il est nécessaire de vérifier si certains éléments ne sont pas déjà présents dans le triplestore (section 6.3).

Le langage SPARQL ne permet pas de faire des insertions atomiques conditionnelles du type "insérer, seulement si tel élément n'existe pas déjà". On peut donc exécuter une requête ASK avant la requête d'insertion (INSERT), mais il peut y avoir un problème d'accès concurrent. En effet, si deux clients ajoutent une annotation au même moment, il y a un risque qu'ils utilisent le même identifiant pour la zone, l'annotation et la forme. Il faut donc que l'exécution du ASK et du INSERT soit un groupe d'opérations bloquant.

Les notes de l'auteur en annexe (Annexe C.2) offrent un résumé des problématiques d'accès à une ressource critique et de l'utilisation de sémaphores en PHP.

La liste des fonctions de **sparql_request_pool.php** et leur description se trouvent dans la documentation technique en annexe (voir respectivement D.2.1 et D.2.2).

6.4.3 Affichage et mise en forme des réponses des requêtes

Le fichier **display_sparql_lib.php** contient toutes les fonctions PHP qui permettent de mettre en forme les résultats des requêtes SPARQL retournés par les fonctions de **sparql_request_pool.php**. Ces fonctions ne sont pas automatiquement appelées, c'est à la charge du développeur d'appeler la bonne fonction.

Pour la plupart des fonctions de **sparql_request_pool.php** il existe une fonction d'affichage correspondante dans **display_sparql_lib.php**. Cette séparation permet de facilement modifier l'affichage d'un résultat de requête sans devoir toucher ni à la structure ni aux requêtes.

La liste des fonctions de **display_sparql_lib.php** et leur description se trouvent dans la documentation technique en annexe (voir respectivement D.3.1 et D.3.2).

6.5 Recherche et navigation dans les archives

6.5.1 Navigation dans les archives

Le modèle de données (présenté au chapitre 4) est très expressif et aussi fidèle qu'il puisse l'être à la manière dont la cote est composée. De simples requêtes SPARQL permettent alors de retourner les différentes étapes de la navigation (l'interface de navigation a été présentée dans le chapitre 5) :

- liste des sections de la BGE
- liste des boîtes de la section
- liste des subdivisions de la boîte
- liste des feuillets de la subdivision
- liste de tous les feuillets de la boîte

Comme ces requêtes sont triviales, on ne détaille ici, à titre illustratif, que l'une d'entre elles : celle qui permet d'afficher toutes les subdivisions d'une boîte. Les autres requêtes peuvent être consultées dans le fichier **sparql_lib_pool.php**.

6.5.1.1 Algorithme pour la navigation

Lors de la navigation, la page **archive_navigation.php** affiche les différentes étapes de la navigation décrites à la section 4.1.3 (sections, boîtes, subdivisions, feuillets) sous la forme de tableaux HTML. L'algorithme (voir listing 6.10) prend comme entrée les paramètres GET. Une analyse de ceux-ci permet de déterminer l'étape de navigation spécifiée par la requête, puis d'afficher une liste de résultats (hyperliens) dans un tableau HTML (sortie de l'algorithme). La fonction **getSubdivisionFromBox** (voir listing 6.11) génère, puis exécute la requête SPARQL. Son retour est ensuite mis en forme avec la fonction **printArrayAsTable**.
N.B. : C'est la présence du paramètre qui est importante, car son nom est utilisé dans les tests (sa valeur n'a pas d'importance).

Le listing 6.10 résume le test des paramètres, mais sans le détail. Seul l'algorithme pour le paramètre "box" est détaillé ici à titre illustratif, car les autres fonctionnent sur le même principe.

```
if ( array_key_exists('all_in_box', $_GET) ) {
} elseif ( array_key_exists('section', $_GET) ) {
} elseif ( array_key_exists('subdivisions', $_GET) ) {
    if ( ( array_key_exists('box', $_GET) ) && (strcmp($subdivisions,"empty") == 0) ) {
    } else { }
} elseif ( array_key_exists('box', $_GET) ) {
        $box = $_REQUEST["box"];
        echo "<h4>".$IIPlang['BROWSE_BOX_DIVS']." : ". htmlspecialchars($box) ." </h4>";
        $tttemp = getSubdivisionFromBox($prefixes, $graph_file, $box);
        $resultQueryArray = json_decode($tttemp, true);
        printArrayAsTable($resultQueryArray, "NavigationBox",
                          $prefixes, $box, "", "", "", "", "", $storage);
} else { }
```

Listing 6.10 – Test des paramètres dans archive_navigation.php

```
function getSubdivisionFromBox($prefixes, $graph_file, $box) {
   $sparql_mode = "query";
   global $evaluationQueriesEndpoint;
   $query =
         $prefixes . "
         SELECT DISTINCT (count(?feui) as ?rawFeuillet) ?Subdivision
         FROM " . $graph_file . "
         WHERE {
            {
            <$box> p:hasSurfaceEcriture ?feui .
            }
            UNION
            {
            <$box> p:hasSubdivisions ?Subdivision .
            ?Subdivision p:hasSurfaceEcriture ?feuillet .
            }
         } GROUP BY ?feui ?Subdivision ORDER BY ?Subdivision
      ";
   return sparql_curl_request($sparql_mode, $evaluationQueriesEndpoint, $query);
}
```

Listing 6.11 – Fonction getSubdivisionFromBox

6.5.2 Moteur de recherche

Cette section décrit le fonctionnement et les limitations du moteur de recherche développé pour ce projet. Les trois formulaires (présentés dans la section 5.2.2) permettent de faire des recherches par concept, par cote ou par contenu. Leur principe de fonctionnement étant identique, seul le moteur de recherche par contenu est détaillé ici, car c'est le plus complexe des trois.

6.5.2.1 Algorithme de recherche

Lors de la recherche, la page **search_results.php** affiche les résultats de recherche sous la forme de tableaux HTML. L'algorithme (voir listing 6.12) prend comme entrée les paramètres POST. Une analyse de ceux-ci permet de déterminer le type de recherche (par cote, par concept, par contenu textuel) et les options de recherche (mot entier, etc.), puis d'afficher la liste des résultats (hyperliens vers l'image et texte avec le/les mots-clefs en évidence) sous la forme d'un tableau HTML (sortie de l'algorithme). La fonction **searchAnnotationOr_what** génère, puis exécute la requête SPARQL (voir listing 6.13). Son retour est ensuite mis en forme avec la fonction **printSearchResults**.
N.B. : le type de recherche est déterminé en testant l'existence des paramètres "SEARCH_COTE", "SEARCH_CONCEPT" et "SEARCH_ANNOTATIONS_OR" (peu importe leur valeur). Les autres paramètres sont propres à chaque type de recherche et peuvent être consultés dans le fichier **sparql_lib_pool.php**.

Le listing 6.12 résume le test des paramètres, mais sans le détail. Seul l'algorithme pour le paramètre "SEARCH_ANNOTATIONS_OR" est détaillé ici à titre illustratif, car les autres fonctionnent sur le même principe.

```
if ( array_key_exists('ACTION', $_POST) ) {
  $action = $_REQUEST["ACTION"];
  switch ($action) {
    case "QUERY":
       $sparql_mode = "query";
       if ( array_key_exists('SEARCH_COTE',$_POST) && array_key_exists('imagesfilename',$_POST) ){
       } else if ( array_key_exists('SEARCH_ANNOTATIONS_OR', $_POST)
                 && array_key_exists('imagesfilename', $_POST) ) {
          // Set local variables from POST
          // Trim pour eviter de rechercher un caractere vide (== retourne tout !)
          $searchWords = trim($_REQUEST["imagesfilename"]); // TODO renommer imagefilename
          $searchfor = $_REQUEST["SEARCHFOR"];
          $searchbool = $_REQUEST["SEARCHBOOL"]; // AND/OR

          $matchWhole = "LOOSE"; // set the default value
          if (isset( $_REQUEST["MATCH_WHOLE"] )) {
             // Overwrite if set
             $matchWhole = $_REQUEST["MATCH_WHOLE"];
          }

          $mode_regex = "";
          if ( array_key_exists('MODE_REGEX', $_REQUEST) ) {
             $mode_regex = $_REQUEST["MODE_REGEX"];
          }

          // Split The Search Words, remove Duplicates and sort by string length (longest first)
          $searchWordsArray = prepareSearchWordsArray($searchWords, $mode_regex);
```

```
        $resultArray = searchAnnotationOr_what($searchWordsArray, $searchfor,
                        $searchbool, $matchWhole, $mode_regex, $prefixes, $sparql_mode);

        $customText = "<p>".$IIPlang['CUSTOM_TEXT_SEARCH_TERMS']." : </p>";
        printSearchResults($resultArray[0], $searchWordsArray,
                        $resultArray[1], $matchWhole, $customText);
      } else if ( array_key_exists('SEARCH_CONCEPT', $_POST)
              && array_key_exists('concept', $_POST) ) {
      }
      break;
    default:
      echo "NO ACTION DEFINED OR MALFORMED ACTION";
      break;
 } // END SWITCH
}
```

Listing 6.12 – Test des paramètres dans search_results.php

```
function searchAnnotationOr_what($searchWordsArray, $searchfor, $searchbool,
  $matchWhole, $mode_regex , $prefixes, $sparql_mode ) {
   global $evaluationQueriesEndpoint;
   $graph_file = "<file://test_001_2013-04-08.xml>";
   $query = $prefixes . "

SELECT DISTINCT ?Type ?Filename ?Text
FROM " . $graph_file . "
WHERE {
 {
   ?SurfaceCouverte p:contient ?Zone ;
      p:hasFilename ?Filename .
";
   if ( strcmp($searchfor,"annotations") == 0) { // si vrai = 0
      $query .= ' ?Zone p:hasAnnotation ?EDT.';
   } elseif ( strcmp($searchfor,"transcriptions") == 0) { // si vrai = 0
      $query .= ' ?Zone p:hasTranscriptionElement ?EDT.';
   } elseif ( strcmp($searchfor,"both") == 0) { // si vrai = 0
      $query .= ' ?Zone ?hasEDTOrAnnotation ?EDT.';
   }
/* Return raw text */
   $query .= "
   ?EDT rdf:type ?Type;
        p:rawText ?Text .
   FILTER (
";
   // Replace characters that can cause problem in the regex
   $searchWordsArray = prefixForRegexInSPARQL($searchWordsArray, $mode_regex);

   $booleanOperator = " || "; // OR by default
   if ( strcmp($searchbool,"AND") == 0 ) { // si vrai = 0
      $booleanOperator = " && ";
   }
   $booleanOperator .= "\n";

   foreach ($searchWordsArray as $k => $searchTerm) {
      if ( $k > 0) {
         $query .= $booleanOperator;
      }
      if ( strcmp($matchWhole,"WHOLE") == 0 ) { // si vrai = 0
```

```
            $query .=
" REGEX(STR(?Text), \"((?<!\\\\p{L}|\\\\p{N})$searchTerm(?!\\\\p{L}|\\\\p{N}))\", \"iu\") ";
        } else {
            $query .=
' REGEX(STR(?Text), "' . $searchTerm . '", "iu") ';
        }
    }
    $query .= "
    )
 }
} ORDER BY ?Filename ";

    // If the $searchWordsArray is empty, return empty string and do not
    // execute the query to avoid an expected MALFORMED QUERY
    if ( sizeof($searchWordsArray) == 0 ) {
        $resultQuery = "";
    } else {
        $resultQuery = sparql_curl_request($sparql_mode, $evaluationQueriesEndpoint, $query);
    }
    return array($resultQuery, $query) ;
}
```

Listing 6.13 – Fonction searchAnnotationOr_what

6.5.2.2 Fonctionnement du moteur de recherche

Comme on l'a vu dans l'état de l'art (voir section 3.3.3.2), les solutions basées sur des index (Lucene...) ont été écartées, pour privilégier une méthode plus simple et moins intrusive. Le moteur de recherche de ce projet utilise la puissance et la flexibilité des requêtes SPARQL, avec un filtre par expression régulière (regex).

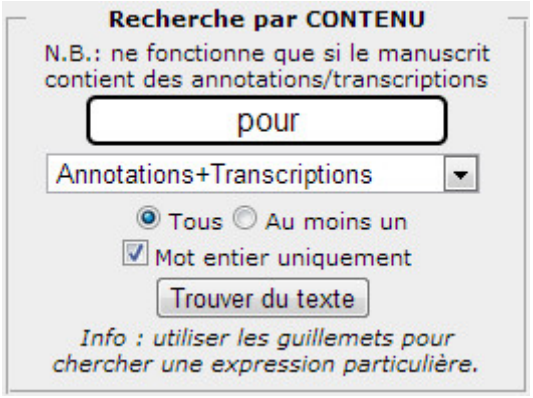


FIGURE 6.5 – *Formulaire de recherche (par contenu)*

La saisie des termes de recherche se fait dans un formulaire HTML (voir la figure 6.5 et la section 5). Le champ principal (*input* de type *text*) contient le terme de recherche "pour", tandis qu'une liste déroulante (*select*) permet de choisir si la recherche s'effectue dans les **annotations**, les **transcriptions** ou **les deux** simultanément. Un bouton *radio* permet de choisir si les résultats doivent contenir **Tous** ou **Au moins un** des termes de recherche. La dernière option est une case à cocher (*checkbox*) permettant de chercher une expression particulière si les termes sont entre guillemets (par ex. : plusieurs mots, y compris les espaces).

Le formulaire envoie les paramètres en POST sur la page **search_results.php**. Celle-ci communique avec le *backend* (section 6.4) pour la construction de la requête SPARQL (décrite ci-dessous) et l'affichage des résultats.

Le listing 6.14 correspond à la requête de recherche du mot "pour" dans les annotations. La requête SPARQL permet d'extraire les données dans lesquelles effectuer la recherche, alors que l'expression régulière permet de filtrer les résultats selon leur contenu.

```
PREFIX rdfs:<http://www.w3.org/2000/01/rdf-schema#>
PREFIX rdf:<http://www.w3.org/1999/02/22-rdf-syntax-ns#>
PREFIX p:<http://saussure.com/property/>
PREFIX saussure:<http://saussure.com/ressource/>

SELECT DISTINCT ?Type ?Filename ?Text
FROM <file://test_001_2013-04-08.xml>
WHERE{ {
   ?SurfaceCouverte p:contient ?Zone ;
                                    p:hasFilename ?Filename .
   ?Zone p:hasAnnotation ?Annot.
   ?Annot rdf:type ?Type;
                p:rawText ?Text .
   FILTER (
         REGEX(STR(?Text), "pour", "iu")
   )
} } ORDER BY ?Filename
```

Listing 6.14 – SPARQL

Concrètement, la recherche se fait sur le contenu textuel (littéral RDF) de l'ensemble des annotations et/ou des éléments de transcription ; l'expression régulière est appliquée à chacun de ces littéraux, et s'il y a un *match*, alors le contenu de l'annotation ou de la transcription fera partie des résultats.

Cette méthode présente le désavantage évident que les recherches risquent de devenir plus lentes à retourner un résultat, au fur et à mesure que le contenu du triplestore s'étoffera. En revanche, l'avantage est que les résultats sont en permanence à jour.

Cette requête (6.14) cherche toutes les SurfaceCouvertes (voir la terminologie utilisée dans le diagramme UML 4.2) qui contiennent une Zone et qui sont liées à un nom de fichier, et dont les Zones ont une annotation, et dont l'annotation a un type et est liée à un texte. Si le texte *match* l'expression régulière du filtre, alors la requête retourne le Type (annotation), le nom de fichier et le texte. S'il y a plusieurs *matchs* dans un seul texte, il n'est retourné qu'une seule fois.

La requête SELECT ne nécessite pas d'autres explications, sauf concernant les différences induites par les options de recherche et le filtre, qui n'est pas une option courante.

Le prédicat lié à la Zone change selon si l'on cherche dans les **annotations**, les **transcriptions** ou **les deux** simultanément :

Dans les annotations :

```
   ?Zone p:hasAnnotation ?Annot.
```

Dans les transcriptions :

```
   ?Zone p:hasTranscriptionElement ?EDT.
```

Dans les deux :

```
   ?Zone ?hasEDTOrAnnotation ?EDT.
```

On remarque que, grâce au modèle proposé (voir la section 4.2), la valeur du prédicat suffit à différencier une annotation d'un élément de transcription. Ils sont cependant typés (`rdf:type`) pour une meilleure cohérence de l'information (voir la section 4.3).

Les autres modifications interviennent dans le filtre par regex [9]. Celui-ci prend en paramètres deux littéraux (un *text* et un *pattern*), et un troisième littéral optionnel (*flags*).

```
FILTER ( REGEX(STR(?Text), "pour", "iu") )
```

Cet exemple illustre la recherche d'un simple mot dans les annotations. La requête va retourner toutes les annotations dont le texte contient par exemple "pour", "Pourquoi" ou "Singapour".

Ici, le texte (la variable `?Text`) est le contenu d'une annotation ou d'un élément de transcription ; il est *casté* en *string* pour éviter les mauvaises surprises. L'expression régulière est simplement le terme de recherche ("pour"), et les options sont "iu", respectivement pour *insensitive* (insensible à la casse) et *Unicode* (regex en mode Unicode).

Lorsqu'il y a plusieurs termes de recherche, plusieurs regex sont combinées avec des opérateurs booléens :

Recherche de **Tous** les termes (sans "Mot entier") : avec opérateur booléen AND (&&)

```
FILTER(REGEX(STR(?Text), "mais", "iu")  &&
       REGEX(STR(?Text), "pour", "iu") )
```

Recherche d'**au moins l'un** des termes (sans "Mot entier") : avec opérateur booléen OR (||)

```
FILTER(REGEX(STR(?Text), "mais", "iu") ||
       REGEX(STR(?Text), "pour", "iu") )
```

Recherche de **Tous** les termes, avec "**Mot entier**" : avec regex Unicode pour *word boundaries*

```
FILTER(REGEX(STR(?Text), "((?<!\\p{L}|\\p{N})mais(?!\\p{L}|\\p{N}))", "iu") &&
       REGEX(STR(?Text), "((?<!\\p{L}|\\p{N})pour(?!\\p{L}|\\p{N}))", "iu") )
```

6.5.2.3 Problème de *word boundaries* en Unicode

Le *flag* Unicode (*u*) est important, car le texte est un littéral stocké en UTF-8. Il peut donc contenir des caractères sanskrits, cyrilliques, etc. Certains caractères latins accentués sont également encodés différemment que dans des normes *single-byte* (par ex. : ISO-8859-1). Le fait que le contenu soit encodé en UTF-8 a également un impact sur la délimitation des mots (*word boundaries*). En effet, ce qui détermine ce qu'est un mot dans une expression régulière standard n'est pas pareil en Unicode [10]. Comme on peut le voir dans le dernier exemple de filtre ci-dessus, la recherche avec "**Mot entier**" nécessite de délimiter les termes avec des *shorthands* Unicode (voir ci-dessous).

Avec une regex standard, le caractère spécial `\b` (*word boundaries*) sert à délimiter les mots. Dans certains moteurs de regex, le caractère `\b` reconnaît les lettres et les chiffres Unicode. Dans d'autres, comme PCRE (moteur de regex de Perl, utilisé aussi par PHP), le caractère `\b` reconnait seulement les lettres et chiffres "ascii" [11]. C'est-à-dire qu'il est par exemple impossible d'isoler un mot en sanskrit lorsque \b n'est pas *Unicode-aware*. Il n'existe pas d'équivalent Unicode au \b, mais on arrive à un résultat le plus souvent satisfaisant avec les *shorthands*

9. http://www.w3.org/TR/rdf-sparql-query/#funcex-regex
10. http://www.regular-expressions.info/wordboundaries.html
11. http://www.regular-expressions.info/wordboundaries.html

Unicode [12] (\p{L}, \p{N} etc.) et les *lookarounds* [13].
N.B. : Pour que cela fonctionne, il faut donc bien régler l'encodage au niveau du code PHP (mb_internal_encoding + mb_regex_encoding).

Ainsi, dans un contexte Unicode en PHP [14], l'expression :

```
\\\\b$searchTerm\\\\b
```

devient (avec *lookarounds* pour "tout ce qui n'est pas une lettre") :

```
((?<!\\\\p{L})$searchTerm(?!\\\\p{L}))
```

Un exemple de recherche Unicode et son résultat (figures 6.6 et 6.7) indiquent que le système fonctionne bien avec les annotations en UTF-8, même si la recherche ou l'annotation contient plusieurs jeux de caractères.

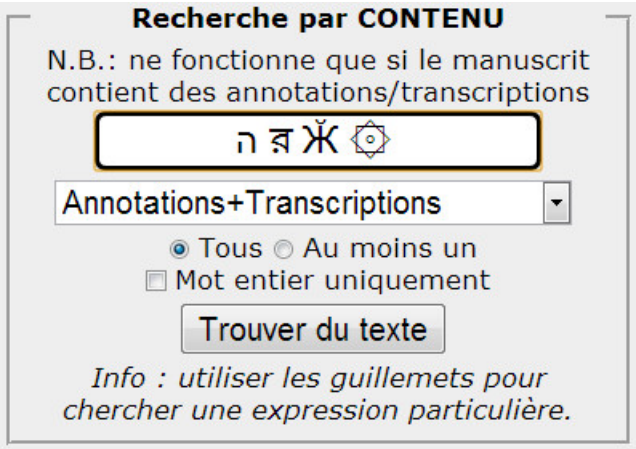


FIGURE 6.6 – *Recherche de quatre caractères UTF-8 dans quatre jeux de caractères différents*

6.5.2.4 Regex et HTML

En réalité, la recherche dans du HTML n'est pas triviale. Il est communément admis que les regex ne sont pas adaptées pour faire de la recherche dans du HTML. En effet, plusieurs problèmes peuvent survenir :

- Faux positifs si le terme de recherche est contenu dans la balise ou ses attributs.
  Par ex. : la recherche de "`di`" retourne "`<div>Texte de test pour l'exemple</div>`"
- Faux positifs si le terme de recherche est contenu dans une entité HTML.
  Par ex. : la recherche de "`copy`" retourne "`<p>Author name ©</p>`"
- Faux négatifs si un mot est partiellement découpé par des balises.
  Par ex. : la recherche de "partiellement" ne retourne pas "`<p>Mot parti<em>elle</em>ment en italique.</p>`"

Pour retourner des résultats corrects, il faudrait donc pouvoir ignorer les balises (et leurs attributs), ainsi que les entités HTML. Or, une expression régulière ne peut pas remplir ces deux tâches, du moins pas avec un taux de réussite de 100%.

Si le code HTML ne présente aucune erreur de syntaxe (aucune balise non fermée, etc.), il est possible d'écrire une expression pour ignorer les balises, mais pas pour exclure les entités.

12. `http://www.regular-expressions.info/unicode.html`
13. `http://stackoverflow.com/questions/2432868/php-regex-word-boundary-matching-in-utf-8`
14. Exemple en PHP, d'où le nombre de *backslashes*.

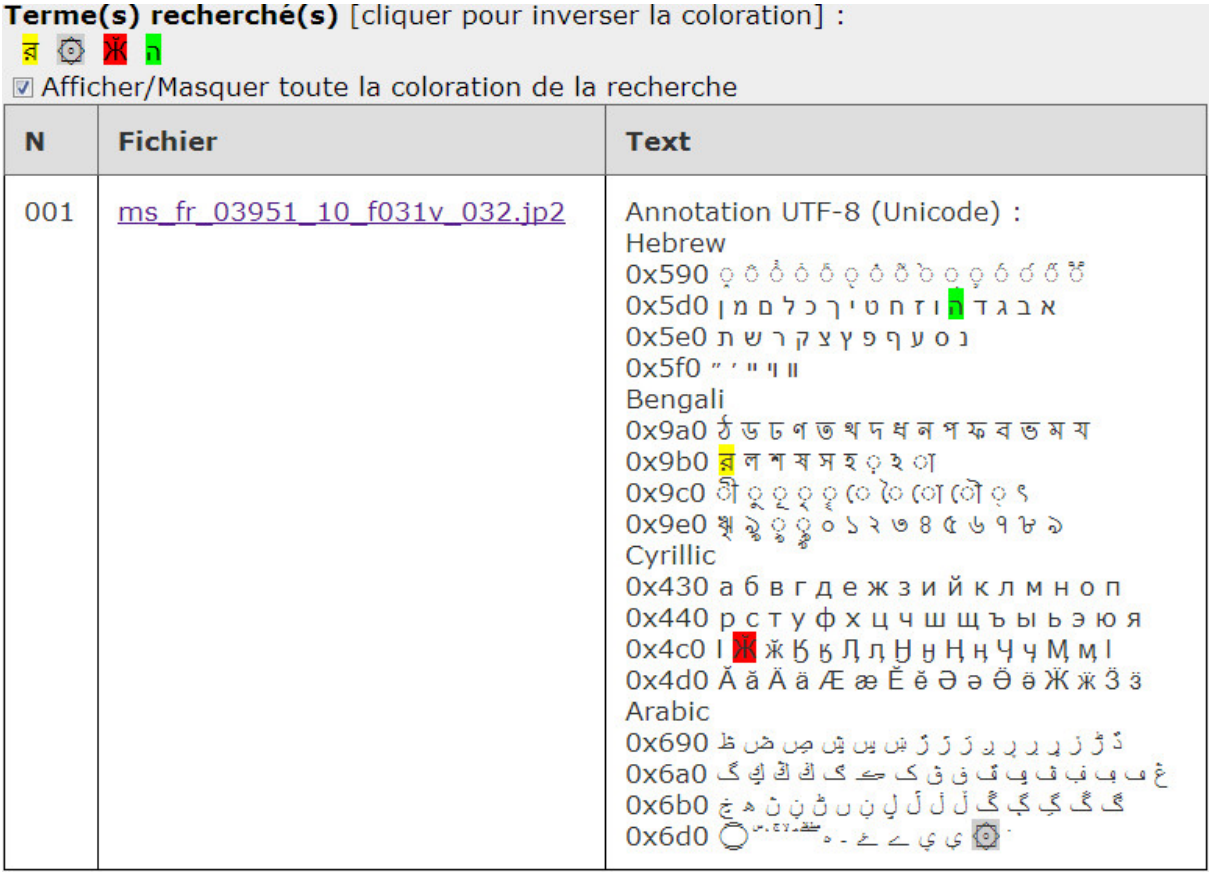


FIGURE 6.7 – *Résultat d'une recherche mélangeant quatre jeux de caractères différents*

Elles pourraient être converties (décodées) avec un traitement préalable, avant l'exécution de la regex. Comme la fonction `REGEX` fait partie intégrante de la requête SPARQL, aucun traitement préalable n'est possible.

Le moyen le plus simple de contourner ce problème consiste à stocker dans le triplestore une information sémantiquement redondante, mais syntaxiquement différente pour faciliter la recherche.

```
?Annot p:texte "<div>Texte <em>pour</em> l'exemple &amp; l'illustration</div>";
       p:rawText "Texte pour l'exemple & l'illustration" .
```

Ainsi, lors de l'ajout ou de la modification d'un texte, deux versions sont sauvegardées : une version formatée en HTML pour l'affichage et une version en texte brut (*raw*) pour la recherche. Pour cette dernière, les balises HTML sont supprimées et les entités décodées. Le désavantage de cette méthode est la perte de la mise en forme, mais celle-ci n'est de toute façon pas essentielle pour de la recherche textuelle.
N.B. : Pour permettre des recherches en ignorant les accents (par exemple, que la recherche de "etre" retourne aussi "être"), on pourrait également stocker une troisième version du texte dans laquelle les accents de la version brute seraient supprimés.

6.5.2.5 Coloration des termes de recherche

Afin de visualiser plus facilement les termes de recherche dans les résultats, il est possible d'utiliser une coloration des mots-clefs.

Les résultats sont obtenus grâce à la requête SPARQL présentée précédemment. Elle peut retourner le texte brut ou le texte formaté en HTML. La coloration des termes de recherche est effectuée sur les résultats retournés par la requête.

Comme les résultats sont présentés dans une page HTML, la coloration consiste à entourer les *matchs* par des balises de mise en forme (`mark`, `span` avec une class css...). Il faut donc insérer des balises sans casser la structure du document HTML.

Une méthode naïve consisterait à simplement remplacer les occurrences de `terme` par `<mark>terme</mark>`. Si le terme fait partie d'une balise, la structure HTML sera cassée et affichera un résultat incorrect. Par exemple, la recherche du terme "di" dans :

```
<div>Saussure dit signifié</div>
```

Résultat (source) :

```
<<mark>di</mark>v>Saussure <mark>di</mark>t signifié</<mark>di</mark>v>
```

Résultat obtenu (affiché) : <**di**v>Saussure **di**t signifiédiv>
Résultat souhaité (affiché) : Saussure **di**t signifié

La majorité des méthodes courantes se font du côté client (Javascript) et utilisent une regex sur les nœuds textuels du DOM, ce qui évite les problèmes des balises et des entités. Or, pour d'obscures raisons, le moteur de regex de Javascript n'est pas *Unicode-aware*, alors que l'encodage par défaut du langage est l'UTF-8. Ceci rend impossible la détection de caractères Unicode (non ascii).

Comme la coloration est un luxe, on pourrait se contenter d'une solution Javascript offrant une coloration basique limitée aux caractères ascii. Cependant, le résultat d'un bref sondage informel a révélé qu'il est préférable de n'avoir aucune coloration, plutôt qu'une coloration valable pour certains termes de recherche seulement. Dès lors, les solutions Javascript ne conviennent pas à nos besoins. Il convient donc d'appliquer la coloration au niveau du serveur, car le moteur de regex de PHP est *Unicode-aware* contrairement à celui de Javascript.

Le moteur de regex du langage PHP [15] (et par conséquent toutes les fonctions "preg") est basé sur celui de Perl : PCRE (Perl-compatible regular expressions) [16]. Pour accepter l'UTF-8, le moteur PCRE doit être compilé avec l'option "–enable-unicode-properties". On peut vérifier si c'est le cas avec la fonction `phpinfo()` qui devrait retourner :

```
PCRE (Perl Compatible Regular Expressions) Support enabled
PCRE Library Version 8.02 2010-03-19
```

Dans la partie "Regex et HTML", on a vu que les regex sont mal adaptées à l'HTML, et on a contourné ce problème en faisant la recherche sur le texte brut. La coloration au niveau du serveur pose les mêmes problèmes avec les balises et les entités HTML que pour la recherche.

Les possibilités sont donc :

- Solution 1 : utiliser une regex qui ignore les balises HTML, et remplacement dans l'HTML des mots-clefs avec "preg_replace".
  + Permet la recherche par regex en UTF-8, insensible à la casse.
  - N'ignore pas les entités.
  - Ne fonctionne que si le code HTML n'a aucune erreur de syntaxe.
- Solution 2 : utiliser la version en texte brut (plutôt qu'en HTML) pour l'affichage des résultats de recherche.
  + Permet la recherche par regex en UTF-8, insensible à la casse, aucun problème de balises ou d'entités.
  + Permet de garder la même requête que dans le filtre SPARQL
  - Perte de la structure (paragraphes, etc.).

15. `http://www.regular-expressions.info/php.html`
16. `http://www.regular-expressions.info/pcre.html`

- Solution 3 : charger le HTML en DOM et traiter les nœuds "textuels".
  + Permet d'ignorer les balises HTML tout en gardant la structure du document.
  - Utilisation d'une requête XPath (fonction "contains"). La fonction "contains" n'est pas *insensitive*, sauf en utilisant Translate (ne marche que pour les caractères latins non accentués).
  - Difficulté de détecter les *matchs* coupés par des balises (par ex. : difficile de *match* le mot "partiellement" : `<p>Texte avec mot parti<em>ellement</em> en italique</p>`).

En utilisant une combinaison des solutions 2 et 3, on arrive à un résultat satisfaisant. La solution finalement la plus intuitive pour l'utilisateur est de lui retourner le texte brut. Comme il s'agit toujours d'un affichage en HTML, on ajoute le contenu du texte brut dans une balise HTML (p ou div). Cela libère le contenu des balises "problématiques" (balises au milieu du mot). Ce texte est chargé dans un DOM et les balises de mise en forme sont insérées dans le DOM avec XPath. Cette opération est séquentielle, car les termes de recherche sont traités les uns à la suite des autres. Dès lors, il existe à nouveau le risque de ne pas détecter un terme si celui-ci est découpé par une balise fraîchement ajoutée. Ce problème est réglé en triant les termes de recherche par longueur décroissante. Ainsi, il n'y a aucun risque de casser une balise HTML, comme un mot plus long sera toujours traité avant un mot plus court.

Afin que les différents termes de recherche soient bien visibles, on utilise une couleur différente pour chaque terme (figure 6.10). L'utilisation de la balise `mark` n'est donc pas possible, on utilise alors des `span` avec des classes CSS. Il y a quinze classes (`highlight1` à `highlight15`), et s'il y a plus de quinze termes de recherche, la coloration recommence dans le même ordre (ce qui fait que les termes 1 et 16 auront la même couleur, celle définie par `highlight1`). Si plusieurs des termes de recherche sont contenus dans un mot, les colorations se chevauchent. Finalement, comme la lecture peut vite devenir pénible s'il y a beaucoup de termes de recherche, il est possible de désactiver la coloration pour tous les termes, ou certains d'entre eux, en cliquant sur le terme dans la liste au-dessus des résultats (voir la figure 6.11) .

Un exemple complet illustre les problématiques traitées dans cette section. Le formulaire de la figure 6.8 montre les termes de recherche utilisés pour une recherche Unicode. Il s'agit de quatre chaînes de caractères en UTF-8, dans quatre jeux de caractères différents (bengali, hébreu, arabe et cyrillique) [17]. La figure 6.9 est la requête exacte correspondant à cette recherche. La figure 6.10 montre le résultat de cette recherche, alors que la figure 6.11, illustre le même résultat, mais quand la coloration de l'un des termes de recherche a été manuellement désactivée par l'utilisateur.

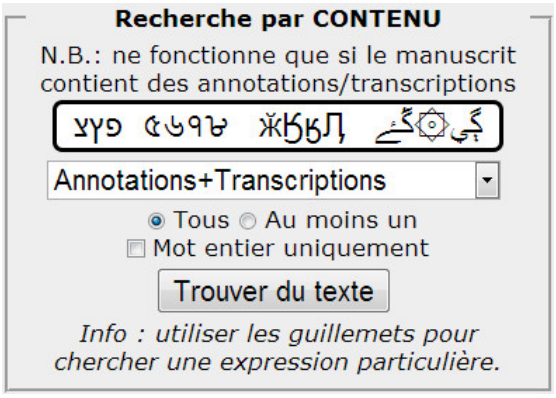


FIGURE 6.8 – *Recherche de quatre suites de caractères dans quatre jeux de caractères UTF-8*

17. Les chaînes de caractères sont des groupes de caractères complètement aléatoires. L'auteur s'excuse d'avance si elles ont une quelconque signification, surtout si celle-ci est grossière ou blessante.

```
SELECT DISTINCT ?Type ?Filename ?Text
FROM <file://test_001_2013-04-08.xml>
WHERE {
 {
   ?SurfaceCouverte p:contient ?Zone ;
                    p:hasFilename ?Filename .
   ?Zone ?hasEDTOrAnnotation ?EDT.
   ?EDT rdf:type ?Type;
         p:rawText ?Text .
   FILTER (
     REGEX(STR(?Text), "৫৬৭৮", "iu")  &&
     REGEX(STR(?Text), "ۓ۞ڲڴ", "iu")  &&
     REGEX(STR(?Text), "ӂҔҕԒ", "iu")  &&
     REGEX(STR(?Text), "פץצ", "iu")
   )
 }
} ORDER BY ?Filename
```

FIGURE 6.9 – *Requête SPARQL de quatre suites de caractères dans quatre jeux de caractères UTF-8*

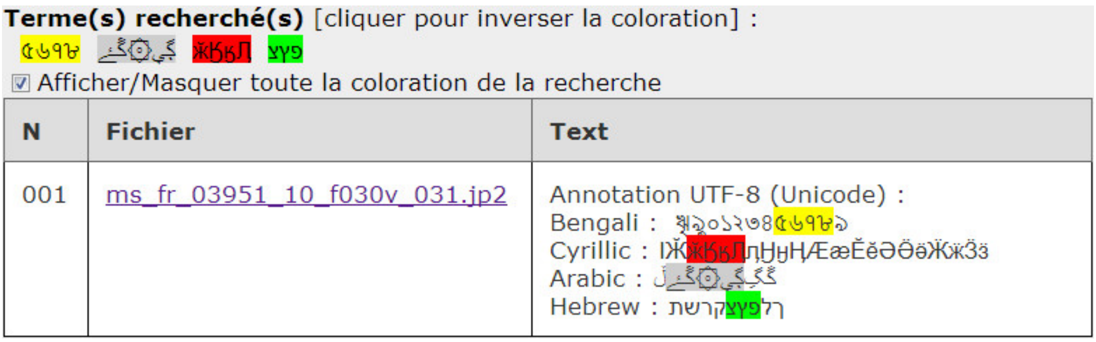


FIGURE 6.10 – *Résultat de recherche (tous les termes de recherche colorés)*

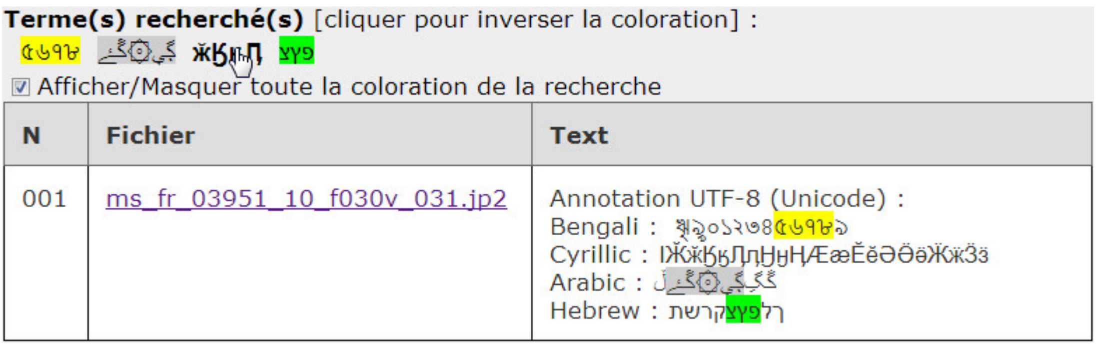


FIGURE 6.11 – *Résultat de recherche (l'un des termes de recherche non coloré)*

6.6 Application client/serveur - Service frontal (PHP)

Le service frontal est un script PHP (**annotations.php**). Il reçoit des requêtes de l'application client qui sont traitées, puis transférées au *backend* avant de renvoyer une réponse au client. Ce service offre les fonctionnalités suivantes :

- Récupérer la liste des annotations, des annotations conceptuelles ou des éléments de transcription.
- Récupérer la liste des images liées au manuscrit courant (liste des *thumbnails*).
- Créer une annotation, une annotation conceptuelle ou un élément de transcription.
- Éditer/modifier une annotation, une annotation conceptuelle ou un élément de transcription.
- Supprimer une annotation, une annotation conceptuelle ou un élément de transcription.

L'exécution de chacune de ces actions est initiée par l'application client sous la forme d'une requête HTTP envoyée au script **annotations.php**. Celui-ci agit comme un service web, car l'information qu'il renvoie est destinée à être interprétée par le client, contrairement aux scripts **archive_navigation.php** et **search_results.php** qui affichent directement le résultat en tant que page web.

Les paramètres GET/POST contiennent l'information nécessaire à l'exécution de la fonction appropriée sur le serveur. On peut donc considérer ce service comme un service REST.

Les fonctionnalités du service sont accédées avec le paramètre ACTION. Il existe quatre actions :

- **QUERY** Pour récupérer les différentes listes.
- **ADD** Pour les ajouts.
- **EDIT** Pour les éditions/modifications.
- **DEL** Pour les suppressions.

Chacune des actions est accompagnée de paramètres bien précis. Par exemple, pour récupérer une liste, il sera nécessaire de fournir le nom de l'image courante ; pour créer une annotation, il sera nécessaire de fournir le nom de l'image courante, ainsi que les coordonnées d'une zone et le texte qui lui est associé, etc.

À la réception de ces requêtes, le script **annotations.php** appelle les fonctions correspondantes à l'action dans les fichiers du *backend*. Comme celui-ci a été évoqué plus tôt (voir la section 6.4), on ne décrit ici que les échanges et la manière dont le service frontal les gère.

6.6.1 Algorithme du service web (frontal)

Lorsque le service web est appelé, le script **annotations.php** analyse les paramètres POST et retourne une information structurée au client. L'algorithme (voir le listing 6.15) prend comme entrée les paramètres POST. Une analyse de ceux-ci permet de déterminer l'action à effectuer (requête d'information, ajout, modification ou suppression de données) et les options spécifiques à chaque action. L'algorithme retourne une information structurée (et interprétable par l'application client) contenant les résultats (confirmation d'ajout/modification/suppression, liste des annotations, listes de *thumbnails*). Les résultats sont retournés en JSON pour les données et sous forme de texte pour les messages. Au besoin, c'est le client (JavaScript) qui gère l'affichage de ces informations.
N.B. : le type d'action à effectuer est déterminé en testant l'existence et la valeur du paramètre "ACTION" dans les paramètres POST.

À l'exécution du script **annotations.php**, celui-ci vérifie parmi les paramètres POST la présence d'un paramètre **ACTION**. S'il n'existe pas, le script ne fait rien. S'il existe, des actions sont exécutées selon sa valeur (**switch/case**). Dans chaque **case**, un test (**if**) est effectué pour vérifier que les paramètres supplémentaires spécifiques à l'action sont bien définis. Si tel est le cas, alors deux fonctions sont exécutées : une fonction de requête (définie dans **sparql_request_pool.php**) et une fonction de mise en forme (définie dans **display_sparql_lib.php**).
N.B. : Pour chaque fonction de requête, il existe une fonction équivalente pour la mise en forme : **getAnnotationOrTranscription** et **printGetAnnotationOrTranscription**, **editAnnotation** et **printEditAnnotation**, etc.

Toutes les actions sont traitées de la même manière (test des paramètres, requêtes, mise en forme). Le listing 6.15 résume le fonctionnement du script **annotations.php**. Tous les tests sont affichés, mais lorsque plusieurs actions fonctionnent sur le même principe, seul l'algorithme de l'une d'entre elles est détaillé. Par exemple, seul l'ajout d'une annotation est détaillé, car l'ajout d'une transcription ou d'une annotation conceptuelle fonctionne de la même manière.

Pour illustrer le fonctionnement, prenons le cas de l'ajout d'une nouvelle annotation dans le triplestore. La fonction de requête (ici **addNewAnnotation**) ajoute l'annotation dans le triplestore. Le résultat de l'opération est retourné, puis passé en paramètre à la fonction de mise en forme (ici **printAddNewAnnotation**). Celle-ci retourne au client une information dépendante du type de requête. Dans le cas d'une requête de recherche (liste des *thumbnails*, liste des annotations liées à une image, etc.) la réponse contient les résultats encodés en JSON (par ex. : la liste des annotations du manuscrit courant). Dans le cas d'un ajout, d'une édition ou d'une suppression la réponse contient le résultat sous forme de texte (succès/échec de l'opération). Lors d'un ajout, des données supplémentaires accompagnent la réponse. Ces informations permettent au client de mettre à jour ses données locales. Un exemple complet est détaillé dans la section 6.8.

```
if ( array_key_exists('ACTION', $_POST) ) {
   $action = $_REQUEST["ACTION"];
   switch ($action) {
      case "QUERY":
         $sparql_mode = "query";
         $imagesfilename = $_REQUEST["imagesfilename"];
         // SI LES CHAMPS NECESSAIRES SONT DEFINIS, ALORS
         if ( array_key_exists('GET_FROM_FILENAME', $_POST) &&
               array_key_exists('imagesfilename', $_POST) ) {
            $searchfor = $_REQUEST["SEARCHFOR"];
            $resultQuery = getAnnotationOrTranscription($searchfor, $imagesfilename);
            printGetAnnotationOrTranscription($resultQuery);
         } else if ( array_key_exists('GET_THUMBNAILS_FROM_FILENAME', $_POST) &&
                     array_key_exists('imagesfilename', $_POST) ) {
         }
         break; // END QUERY
      case "ADD":
         if ( array_key_exists('newAnnot', $_POST) &&
               array_key_exists('imagesfilename', $_POST) ) {
            // Set local variables from POST / Decodage du JSON
            $imagesfilename = $_REQUEST["imagesfilename"];
            $resultNewAnnotArray = json_decode($_REQUEST["newAnnot"], true);
            $resultArray = addNewAnnotation($imagesfilename, $resultNewAnnotArray, $prefixes);
            $resultInsert = $resultArray[0];
            $resultQuery = $resultArray[1];
            $jsonReturned = $resultArray[2];
            printAddNewAnnotation($resultInsert, $resultQuery, $jsonReturned);
```

```
        } elseif ( array_key_exists('newTrans',$_POST) &&
                   array_key_exists('imagesfilename',$_POST) ) {
        } elseif ( array_key_exists('newConceptualAnnot',$_POST) &&
                   array_key_exists('imagesfilename',$_POST) ) {
        }
        break; // END ADD
     case "EDIT":
        if ( array_key_exists('annotToEdit', $_POST) ) {
           // Set local variables from POST
           $annotToEdit = $_REQUEST["annotToEdit"];
           $annotToEditArray = json_decode($annotToEdit, true);
           $resultArray = editAnnotation($annotToEditArray);
           $resultEdit = $resultArray[0];
           $resultQuery = $resultArray[1];
           printEditAnnotation($resultEdit, $resultQuery);
        } elseif ( array_key_exists('conceptualAnnotToEdit', $_POST) ) {
        } elseif ( array_key_exists('transElementToEdit', $_POST) ) {
        }
        break; // END EDIT
     case "DEL":
        if ( array_key_exists('annotToDel', $_POST) ) {
           $annotToDel = $_REQUEST["annotToDel"];
           $annotToDelArray = json_decode($annotToDel, true);
           $resultArray = deleteAnnotation($annotToDelArray);
           $resultDelete = $resultArray[0];
           $deleteQuery = $resultArray[1];
           printDeleteAnnotation($resultDelete, $deleteQuery);
        } elseif ( array_key_exists('conceptualAnnotToDel', $_POST) ) {
        } elseif ( array_key_exists('transElementToDel', $_POST) ) {
        }
        break; // END DEL
     default:
        echo "NO ACTION DEFINED OR MALFORMED ACTION";
        break;
  } // END SWITCH
}
```

Listing 6.15 – Extrait du service **annotations.php**

6.7 Application client/serveur - Client (HTML/Javascript)

La section 5 a présenté l'interface utilisateur (GUI) et les interactions qu'elle offre en termes d'affichage d'image et de consultations/éditions de métadonnées (annotation, etc.).

Cette interface est disponible dans la page **viewer_visitor.php**. Lors de l'initialisation de cette dernière, plusieurs modules JavaScript sont téléchargés et vont permettre de rendre la page interactive. Cette application Javascript gère les interactions avec l'utilisateur, ainsi que les communications avec le serveur distant (service frontal décrit dans la section 6.6).

Selon les actions de l'utilisateur, l'application Javascript communique avec le service frontal pour modifier/rafraîchir certaines informations, ou pour ajouter du contenu dynamiquement (annotations, etc.), sans devoir recharger toute la page systématiquement. Cette technique permet de rendre l'interface plus rapide et plus réactive (réduction des transferts et des temps d'affichage).

L'application client communique avec deux services web :
- le serveur IIPImage pour l'affichage des *tiles*.
- le *frontend* (annotations.php) décrit dans la section 6.6.

Le viewer IIPMooViewer s'occupe de charger/d'afficher les *tiles* (protocole IIP), de la navigation sur le manuscrit, de la rotation, etc. Le fonctionnement général du viewer et de ses extensions n'est pas détaillé ici (gestion des *tiles*, rotation, etc.), sauf pour les parties qui ont subi des modifications (annotations-edit-mass.js, etc.).

L'interface offre plusieurs fonctionnalités :
- Navigation, zoom rotation (par défaut dans IIPMooViewer)
- Navigation par *thumbnails*
- La barre d'outils
- Créer (dessiner) une annotation
- Saisie des annotations dans un éditeur externe
- Trois modes permettant de changer le type d'annotation : annotations textuelles, annotations conceptuelles, éléments de transcription
- Affichage des transcriptions disponibles

Le développement du client peut être découpé en deux :
- Modification/création d'extensions du projet IIPMooViewer
  - **La navigation** interagit avec le viewer (recharge l'image et les annotations, etc.)
  - **La barre d'outils** s'intègre dans l'espace du viewer et interagit avec lui
  - **Le système d'annotation graphique** (dessin) et sa gestion
- Mise en place d'une application client personnalisée correspondant à notre système et intégrant les extensions ci-dessus, ainsi que les communications avec le service frontal

6.7.1 Contraintes de développement en Javascript

Les applications web ont la particularité de ne pas se comporter de la même façon d'un navigateur web à un autre. Les applications Javascript ne manquent pas à cette règle. Tout le développement réalisé a été testé avec Mozilla Firefox et Google Chrome. La compatibilité avec les autres navigateurs n'est pas assurée.

Comme IIPMooViewer nécessite le *framework* MooTools [18], on limite la charge de l'application en utilisant également ce *framework* plutôt qu'un second (par ex. : jQuery). Il n'y a donc pas de jQuery dans le code, mais uniquement du Javascript pure ou du code MooTools.

18. `http://mootools.net/`

6.7.2 Liste des fichiers

```
/**************
*** IIPMooviewer (default)
**************/
src/iipmooviewer-2.0.js          // Par défaut - quelques modifications
src/navigation.js
src/protocols/iip.js
src/lang/help.fr.js
src/lang/saussure.fr.js
/**************
*** IIPMooviewer (extensions)
**************/
src/annotations.js               // Par défaut - aucune modifications
src/annotations-edit-mass.js     // Par défaut - beaucoup de modifications
src/extended-toolbar.js
src/navigation_slider.js
/**************
*** Custom files for the application
**************/
src/concepts_saussuriens.js
src/iip_semantic_web_brero.js
script/slideitmoo-1.2-mootools1.3_full_source.js  // Définition de SlideItMoo
src/mass_editor.js
src/addTinyMCE.js
```

Seuls les fichiers qui ont été modifiés (ou créés) sont décrits ici :

- **iipmooviewer-2.0.js** [**modifié**] est la classe principale du viewer. Quelques modifications mineures ont été apportées et sont documentées par la suite.
- **annotations-edit-mass.js** [**modifié/nouveau**] contient l'extension "édition des annotations". Elle se base sur **annotations-edit.js**, mais tellement de modifications ont été appliquées qu'on peut presque le considérer comme un apport complet.
- **extended-toolbar.js** contient l'extension "barre d'outils" pour IIPMooviewer.
- **navigation_slider.js** contient l'extension "carousel/slider" pour IIPMooviewer.
- **concepts_saussuriens.js** contient la liste des concepts et leur URL correspondante.
- **iip_semantic_web_brero.js** contient les fonctions de communication (XMLHttpRequest) avec le service frontal. Ressemble à **sparql_request_pool.php**, mais pour le code Javascript.
- **mass_editor.js** initialise toute l'application Javascript, déclare les variables et les fonctions nécessaires.
- **addTinyMCE.js** contient une fonction pour l'initialisation de l'éditeur WYSIWYG TinyMCE, et contient également une fonction pour le supprimer.

6.7.3 Extensions IIPMooViewer

L'implémentation des fonctionnalités a d'abord été basée sur Lincolnnus/Image-Viewer (voir 3.4.4.4), puis abandonnée pour développer des extensions compatibles avec IIPMooViewer. Trois extensions ont été développées :

- une barre d'outils offrant un accès simple et visuel à plusieurs fonctions du viewer
- un module amélioré d'édition des annotations permettant de dessiner des zones, de les enregistrer de manière personnalisée et d'utiliser un éditeur externe
- un carrousel/navigateur de *thumbnails* pour modifier l'image sans recharger la page

Les extensions reprennent les pratiques de développement utilisées dans IIPMooViewer. Celui-ci utilise le *framework* MooTools [19]. Chaque extension "*implement*" la classe IIPMooViewer. Attention, "*implement*" au sens MooTools permet de modifier une classe existante en lui ajoutant des propriétés ou des fonctions [20].

19. http://mootools.net/
20. http://mootools.net/docs/core/Class/Class#Class:implement

La réutilisation de ces pratiques de développement est importante pour les contributions au projet IIPMooViewer original. En effet, la barre d'outils et le module d'édition d'annotations amélioré sont en cours d'intégration dans le projet original et devraient faire partie de la première version stable d'IIPMooViewer 2.0. Ceci permet de faire profiter la communauté des améliorations développées dans ce travail. Le carrousel, quant à lui, nécessite encore des améliorations avant de pouvoir être proposé comme contribution.

6.7.3.1 Lincolnnus/Image-Viewer

Le projet Lincolnnus/Image-Viewer (voir section 3.4.4.4) utilise IIPMooViewer dans une application permettant notamment de dessiner des annotations et de les sauver dans une base de données relationnelle.

Il a été utilisé comme base de développement dans un premier temps et le troisième prototype d'interface utilisateur (voir la figure 5.4) fonctionnait grâce à lui.

Après plusieurs semaines de laborieuses modifications des sources, et malgré des résultats avancés (dessin, saisie de texte et sauvegarde d'annotations dans le triplestore), il est devenu clair qu'il était préférable de récrire le code plutôt que d'utiliser ces sources. En effet, l'architecture du projet IIPMooViewer permet d'écrire des modules/extensions qui viennent ajouter des fonctionnalités au viewer. Bien que Lincolnnus/Image Viewer ait été réalisé dans cette optique, les sources disponibles ont été partiellement *minifiées* les rendant très difficiles à lire (notamment les sources principales d'IIPMooViewer, contenant des modifications pour la compatibilité de l'application).

Le processus de ***minification*** d'un fichier JavaScript permet de réduire sa taille pour optimiser les performances (transfert et exécution), typiquement en supprimant les commentaires, en raccourcissant les noms de variables, etc. La *minification* peut également être utilisée pour rendre la source illisible et empêcher sa réutilisation.

Comme les auteurs du projet Lincolnnus/Image-Viewer ont apporté des modifications dans les sources d'IIPMooViewer (sans les documenter), cela implique qu'il n'est pas possible d'utiliser ce projet avec les versions plus récentes d'IIPMooviewer, à moins de faire du *reverse engineering* sur le code *minifié*. Rappelons encore une fois qu'IIPMooViewer est en version bêta et qu'il est (et était) donc clairement encore destiné à être modifié.

La réutilisation du projet Lincolnnus/Image-Viewer a été abandonnée et cela a motivé l'écriture de modules propres et compatibles avec IIPMooviewer, destinés à être proposés comme contributions au projet original.

6.7.3.2 extended-toolbar.js

La barre d'outils (**extended-toolbar.js**) a été développée dans ce projet sous la forme d'une extension pour IIPMooViewer. Comme elle a déjà été présentée dans la section 5.2, on ne détaille ici que les fonctionnalités et son intégration dans IIPMooViewer.

Dans cette extension la classe IIPMooViewer "*implement*" une seule fonction : "createExtendedToolbar". Cette fonction prend en paramètre :

- **toolbarId** : identifiant du div HTML qui va contenir la barre d'outils.
- **position** : soit "inside" ou "outside". Par défaut "inside". Permet de spécifier l'emplacement

de la barre d'outils : à l'intérieur (en haut, superposé sur l'image) ou à l'extérieur (le long du viewer en haut ou à gauche, selon le paramètre "orientation").

- **orientation** : soit "vertical" ou "horizontal". Par défaut "horizontal".
- **buttons** : Tableau de *strings* contenant les noms des boutons à ajouter. La liste des *strings* acceptés (et l'action associée à chacun) est la suivante :

0. **fakeDrawNewAnnotation** Bouton *dummy* qui ne fait rien. Il est affiché à la place de "drawNewAnnotation" lorsque l'utilisateur n'est pas authentifié.
1. **drawNewAnnotation** Dessiner une zone et y lier une information. Selon le mode sélectionné, il peut s'agir d'une annotation, d'un élément de transcription ou d'une annotation conceptuelle.
2. **rotateclockwise** Pivoter l'image dans le sens antihoraire (+90˚)
3. **rotateanticlockwise** Pivoter l'image dans le sens horaire (-90˚)
4. **toggleNavigationWindow** Montrer/Cacher la fenêtre de navigation
5. **toggleAnnotations** Montrer/Cacher les zones sur l'image (annotation, etc.)
6. **getRegionalURL** Exporter une image de la vue courante
7. **zoomIn** Zoom avant
8. **zoomOut** Zoom arrière
9. **reset** Réinitialiser la vue
10. **toggleFullscreen** Passer en mode plein écran

N.B. : Ces boutons correspondent aux boutons et à la numération de la figure 6.12.

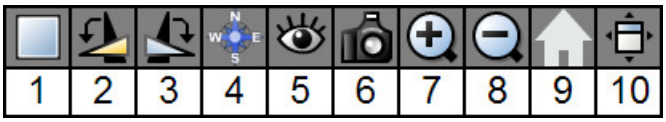


FIGURE 6.12 – *Les boutons de la barre d'outils (extended-toolbar)*

La fonction d'initialisation de la barre d'outils ("createExtendedToolbar") est appelée dans **mass_editor.js**. Le listing 6.16 résume l'initialisation. Elle consiste à tester chaque bouton du tableau "buttons". Si le *string* est correct, alors le bouton est ajouté et l'appel de la fonction correspondante à l'action lui est associé via un événement 'click'. Toutes les autres fonctions appelées sont proposées par défaut dans IIPMooviewer ou ses extensions (généralement accessibles via des raccourcis claviers), hormis la création d'annotations (drawNewAnnotation) qui est décrite dans la section suivante.

```
IIPMooViewer.implement({
  createExtendedToolbar: function(toolbarId, position, orientation, buttons){
...
if (typeof(buttons)==='undefined') buttons = ['rotateanticlockwise', 'rotateclockwise',
  'toggleNavigationWindow', 'getRegionalURL', 'zoomIn', 'zoomOut', 'reset', 'toggleFullscreen'];

  for (var i=0;i<buttons.length;i++) {
    switch (buttons[i]) {
      case 'fakeDrawNewAnnotation':
      case 'drawNewAnnotation':
      case 'rotateclockwise':
        this.rotateclockwisebutton=new Element('img',{'title':SaussureLang['rotateClockwise'],
            'class':'toolButton','src':'images/toolbar/rotate_right.svg'}).inject(toolbar);
        this.rotateclockwisebutton.addEvents({'click':function(){
          _thisIIP.rotate( _thisIIP.view.rotation + 90);
        }});
        break;
      case 'rotateanticlockwise':
      case 'toggleNavigationWindow':
      case 'toggleAnnotations':
      case 'getRegionalURL':
```

```
      case 'zoomIn':
      case 'zoomOut':
        this.zoomInButton=new Element('img',{'title':SaussureLang['zoomOut'],
            'class':'toolButton','src':'images/toolbar/zoom_out.svg'}).inject(toolbar);
        this.zoomInButton.addEvents({'click':function(){
          _thisIIP.zoomOut();
        }});
        break;
      case 'reset':
      case 'toggleFullscreen':
      default:
    }
  }
}
```

Listing 6.16 – Initialisation de la barre d'outils

Les icônes utilisées sont celles du programme Inkscape, sauf "exporter la vue" et cacher la navigation. Les adresses des références sont dans le fichier source.

6.7.3.3 annotations-edit-mass.js

L'extension **annotations-edit-mass.js** étend les fonctionnalités d'édition des annotations dans IIPMooViewer, en offrant une fonction de dessin et en améliorant la manière dont elles sont gérées.
N.B. : À ce niveau du code, une annotation est une zone avec un texte lié. Les notions d'annotations textuelles, conceptuelles et d'éléments de transcription sont toutes regroupées sous l'appellation "annotation".

IIPMooviewer fournit une extension **annotations-edit.js** qui permet de créer ou d'éditer une annotation. Cette extension est fonctionnelle, mais rudimentaire. Plusieurs aspects peuvent être améliorés :

- Lors de la création d'une annotation, une zone rectangulaire apparaît au centre du manuscrit. Pour faire une annotation sur un paragraphe, par exemple, il faut déplacer et redimensionner la zone autour de paragraphe souhaité.
- Un formulaire rudimentaire est attaché à l'annotation. Il reste toujours "accroché" sous la zone, même lorsqu'on la déplace. Impossible de sauver une annotation située en bas de la page, car le formulaire disparaît (sort de la fenêtre).
- La position de la zone, sa taille ainsi que le contenu de l'éditeur sont perdus lorsque la fenêtre est redimensionnée, lorsque l'utilisateur zoome, si l'utilisateur ouvre une autre fenêtre et revient sur la page, etc.
- Il est possible de créer plusieurs annotations simultanément.
- Le système de sauvegarde est générique : pas de différenciation entre un ajout, une édition et une suppression.

Les améliorations de l'extension **annotations-edit-mass.js** sont :

- La création d'une annotation en dessinant une zone rectangulaire englobant la zone souhaitée.
- La possibilité d'utiliser un éditeur externe (comme TinyMCE) à la place du formulaire original "accroché" sous la zone. Il est aussi possible de spécifier un *container* (div) pour contenir l'éditeur et éviter qu'il soit "accroché" à la zone.
- Gestion de la persistance de l'annotation temporaire lors des différents événements (lorsque la fenêtre est redimensionnée, lorsque l'utilisateur zoome, si l'utilisateur ouvre une autre fenêtre et revient sur la page, etc.).
- La création d'une seule annotation à la fois, et du système de contrôle l'accompagnant.
- Différenciation des trois cas : ajout, édition et suppression.

Le fichier original **annotations-edit.js** contient quatre fonctions :

- newAnnotation
- editAnnotation
- updateShape
- updateAnnotations

L'extension **annotations-edit-mass.js** ajoute une cinquième fonction pour le dessin **drawNewAnnotation** et apporte plusieurs modifications aux fonctions **newAnnotation** et **editAnnotation**.

drawNewAnnotation

Dans sa version originale, la fonction **newAnnotation** crée l'annotation (à savoir les coordonnées et un identifiant pour un div temporaire) qu'il injecte dans le canvas du viewer. L'annotation est ensuite passée en paramètre à la fonction d'édition (**editAnnotation**).

N.B. : Attention, lorsqu'on parle de "canvas" dans cette section, on parle du canvas d'IIPMooviewer. **Il ne s'agit pas d'un canvas au sens HTML5**. L'élément div contenant le viewer ( ) contient un autre div qui affiche la portion d'image visible (un groupe de *tiles*) et dont le nom de classe est "canvas". Ce canvas fait la taille de l'image complète à la résolution courante (niveau de zoom), mais il hérite du style "overflow" qui est "hidden". Cela permet d'afficher la portion d'image, tout en gardant la notion de l'image complète.

La fonction **drawNewAnnotation** vient s'ajouter aux fonctions existantes en permettant de dessiner une zone. Lorsque le dessin de la zone est terminé, la zone (les quatre valeurs x,y,w,h) est passée en paramètre à la fonction **newAnnotation**. Cette dernière est modifiée pour prendre un argument "zone" optionnel. Si celui-ci existe, ses valeurs sont utilisées, sinon la zone est créée comme elle l'était dans la version originale.

En ce qui concerne la fonction de dessin, une fois la fonction **drawNewAnnotation** appelée, un canvas temporaire vient se mettre en superposition du canvas du viewer. Des *listeners* pour les événements de la souris (mouseDown, mouseUp et mouseMove) sont ajoutés au canvas temporaire. Sa position est déterminée par les données de style (CSS). Elles sont identiques à celles du canvas original (en tenant compte de l'offset CSS), mais avec un z-index supérieur :

```
canvasTemp.setStyles({'z-index': 1,'left':this.canvas.offsetLeft});
```

Pour dessiner, l'utilisateur clique et garde le bouton enfoncé, déplace la souris pour dessiner la zone et relâche le clic pour terminer.

La zone est un simple div et sa position est déterminée par ses propriétés de style (CSS). Comme l'on souhaite que les zones des annotations soient cohérentes, on utilise des coordonnées relatives à la résolution de l'image (*float* entre 0 et 1). En effet, il est indispensable d'utiliser une telle transformation pour que la zone puisse être reproduite "au bon endroit" à n'importe quel niveau de zoom.

On parle de coordonnées par abus de langage pour faire allusion à la position d'une zone relative à son contexte. En effet, une zone sur l'image sera un div. Sa position et ses dimensions seront déterminées par son style CSS : la position absolue par rapport au haut et à la gauche de la fenêtre (top, left), la largeur et la hauteur. En revanche, les informations sauvées dans le triplestore seront relatives à l'image : un point x,y ainsi qu'une hauteur (h) et une largeur (w).

Les coordonnées sont relatives à la résolution de l'image visualisée (niveau de zoom). A chaque événement (mouseDown, mouseUp et mouseMove) la fonction interne **computeXYWH** est appelée : elle recalcule les valeurs x,y,w,h de la zone (div), puis les convertit en données relatives.

Le recalcul des valeurs x,y,w,h de la zone fait en sorte que la coordonnée (x,y) soit toujours la coordonnée supérieure gauche du rectangle (cas par défaut). Le cas par défaut est un déplacement vers la droite et en bas après avoir cliqué. Le point de départ correspond au clic initial. Comme on peut le voir dans le listing 6.17, dans ce cas x et y ne changent pas, seul la hauteur et la largeur sont mises à jour (**rect** est la zone dessinée (div), **e** est l'événement). Si par contre, on clique et on monte la souris vers la gauche et en haut (par rapport au point de départ), alors on voit que toutes les valeurs sont mises à jour. Les quatre cas sont gérés (de supérieur gauche à inférieur droit, de inférieur gauche à supérieur droit, de supérieur droit à inférieur gauche et inférieur droit à supérieur gauche). Cela permet de dessiner la zone sans restriction.

```
function computeXYWH(rect, e) {
  var a = {};
  //cas de Sup. Gauche vers Inf. Droit (default)
  if ( (e.layerX > rect.lx) && (e.layerY > rect.ly) ) {
    a = {
      x: rect.lx,
      y: rect.ly,
      w: e.layerX - rect.lx,
      h: e.layerY - rect.ly
    }
  //cas de Inf. Droit vers Sup. Gauche
  } else if ( (e.layerX <= rect.lx) && (e.layerY <= rect.ly) ) {
    a = {
      x: e.layerX,
      y: e.layerY,
      w: rect.lx - e.layerX,
      h: rect.ly - e.layerY
    }
  }
```

Listing 6.17 – Extrait de computeXYWH

Une fois que les valeurs de la zone sont déterminées, elles sont converties en données relatives.
Prenons un exemple :
Au niveau de zoom donné, le viewer montre les *tiles* d'une image dont la résolution originale est **3632 x 2413**.

Le canvas d'IIPMooViewer a donc les propriétés :

```
  style="left: -539px; top: -960px; width: 3632px; height: 2413px;"
```

La zone dessinée sur l'image a les propriétés :

```
  style="left: 799px; top: 1167px; width: 82px; height: 23px;"
```

L'annotation a les valeurs :

x : 0.219989, y : 0.483630, w : 0.022577, h : 0.009532

Les valeurs de l'annotation sont relatives à la résolution de l'image. Par exemple, la coordonnée x sera récupérée grâce à l'événement associé à mouseDown :

```
a = {
  x: a.x/_this.canvas.offsetWidth,
  y: a.y/_this.canvas.offsetHeight,
  w: a.w/_this.canvas.offsetWidth,
  h: a.h/_this.canvas.offsetHeight
}
```

Cela correspond aux calculs :

$$x = \frac{799}{3632} = 0.219989 \ , \ y = \frac{1167}{2413} = 0.483630 \ , \ ...$$

Autres améliorations

Les modifications apportées au module d'édition d'annotation sont trop nombreuses pour être détaillées dans ce rapport. On décrit ici brièvement ces ajouts et leurs impacts :

Ajout d'une variable spécifiant si l'instance d'IIPMooViewer est en train de créer une annotation.

```
// Check if the another annotation is currently edited
this.isCreatingAnAnnotation
```

Un argument optionnel "drawnAnnotation" a été ajouté à la fonction **newAnnotation : function(drawnAnnotation)**. Il permet de spécifier une zone (utilisée par drawNewAnnotation au moment où le clic est relâché). Si l'argument n'est pas spécifié, la zone par défaut est générée (au centre de la vue courante).

Un argument optionnel "isNewAnnot" a été ajouté à la fonction **editAnnotation : function(annotation, isNewAnnot)**. Il permet de savoir si l'annotation éditée est une nouvelle annotation ou une annotation déjà existante.

Cela permet de *fire* le bon événement lors d'un ajout : si c'est une nouvelle annotation, c'est l'événement "annotationAdd" qui est *fired*, sinon cela signifie qu'il s'agit d'une édition et c'est l'événement "annotationEdit" qui est *fired* :

```
// If not creating and the annotation is not new => edit
if ( (!_this.isCreatingAnAnnotation) && (isNewAnnot == null) ) {
  _this.fireEvent('annotationEdit', _this.annotations[id]);
} else {
  _this.fireEvent('annotationAdd', _this.annotations[id]);
}
```

Cela permet également de gérer correctement une annulation. Si l'annotation est éditée, l'annulation replace l'annotation locale à sa position et à ses dimensions initiales, et récupère le texte tel qu'il était au moment de l'édition. Si l'annotation est créée, la zone locale temporaire est supprimée.

Ajout de tests pour interdire le dessin d'une annotation lorsque le modulo 360 de l'attribut "IIPMooViewer.view.rotation" est différent de 0 :

```
if (this.view.rotation%360 !== 0)
```

En effet, il y a des problèmes dans IIPMooViewer avec la rotation. Tant qu'ils ne sont pas réglés, il ne sert à rien d'essayer de les gérer pour la création d'annotations.

Un attribut "annotationEditorContainerElement" est un élément destiné à contenir le formulaire. Il doit être spécifié lors de la déclaration du viewer IIPMooViewer (dans ce projet, dans **mass_editor.js**). S'il n'est pas précisé, le formulaire est attaché à la zone.

L'ajout d'un attribut "annotationBeingEdited" permet d'interdire la création de deux annotations simultanément.

Le reste des ajouts concerne pour la plupart la gestion d'un éditeur externe dans le formulaire. Si un éditeur externe est utilisé pour remplacer le formulaire original, un ensemble de fonctions doit être défini :

Pour la création de l'éditeur externe :

```
externalAnnotationEditorCreate();
```

Pour récupérer le contenu de l'annotation (le contenu texte/HTML de l'éditeur externe) :

```
externalAnnotationEditorGetText();
```

Pour récupérer la catégorie de l'annotation :

```
externalAnnotationEditorGetCategory();
```

Pour récupérer le titre de l'annotation :

```
externalAnnotationEditorGetTitle();
```

Pour définir le contenu de l'annotation (le contenu texte/HTML de l'éditeur externe) :

```
externalAnnotationEditorSetHTML()
```

Pour supprimer proprement l'éditeur externe :

```
externalAnnotationEditorSuppress();
```

6.7.3.4 navigation_slider.js

La navigation par *thumbnails* (carrousel/slider) permet de faire défiler les images des manuscrits associés à l'image courante (par exemple, les feuillets d'un même cahier).
Pour ce faire, on utilise le projet SlideItMoo – image slider [21]. Plusieurs autres projets/librairies ont été testés, mais aucun ne correspondait à nos attentes (peu de choix avec MooTools). SlideItMoo est le projet qui s'en approche le plus.
N.B. : Il est très facile de réaliser un carrousel en jQuery, mais comme déjà expliqué plus tôt, l'utilisation de MooTools fait partie des contraintes de développement.

Le fichier **script/slideitmoo-1.2-mootools1.3_full_source.js** contient la libraire SlideItMoo.

Le fichier **src/navigation_slider.js** contient la fonction createThumb.

```
IIPMooViewer.implement({
  createThumb: function(imagepath, currentImageFilename, dataThumb,
    sliderOrientation, navigationBarID1, thumbHeight, DEBUG_MODE){...}
}
```

Listing 6.18 – La fonction createThumb de navigation_slider.js

Celle-ci génère le code HTML adéquat selon l'orientation (paramètre sliderOrientation). En effet, pour correspondre aux classes CSS, les éléments composant le slider ne sont pas identiques s'il est horizontal ou vertical. La fonction se termine par l'instanciation de SlideItMoo.

La génération des balises images est la seule partie qui nécessite d'être détaillée.

Le listing 6.19 montre la portion de code où les *thumbnails* (balises "img") sont générées.

```
var _this = this;
var iipThumbnailParameter = '&JTL=0,0';
var img = new Element('img', {
    'id' : dataThumb[i],
    'data-imageid' : dataThumb[i], // data- HTML5 custom attribute
    'data-imagepath' : imagepath, // data- HTML5 custom attribute
    'src': iipViewer.server+'?FIF='+ imagepath+ '/' + dataThumb[i] + iipThumbnailParameter,
    'alt':dataThumb[i],
    styles:{cursor:'pointer'},
    height:imgHeight
}).inject(temp);

img.addEvent('click', function(){
  _this.setBorderOnSelected(this.id); // Highlight the current image
  _this.fireEvent('thumbnailClicked', this.id);
});
```

Listing 6.19 – Extrait de la fonction createThumb de navigation_slider.js

21. http://www.php-help.ro/php-tutorials/slideitmoo-v11-image-slider/

Le paramètre "dataThumb" est un tableau contenant les noms des fichiers images que le slider doit afficher. Le parcours de ce tableau permet de générer les balises "img" avec les bons attributs. La source ("src") de l'image est une URL de requête vers le serveur IIPImage. On utilise le protocole IIP pour récupérer le *thumbnail*. Le code du listing 6.20 va générer une URL, comme par exemple :
`http://hostname/cgi-bin/iipsrv.fcgi?FIF=/var/www/auto_alim/public_storage/ms_fr_03951_10_f090v_091.jp2&JTL=0,0`

```
'src': iipViewer.server+'?FIF='+ imagepath+ '/' + dataThumb[i] + iipThumbnailParameter,
```

Listing 6.20 – Extrait de la fonction createThumb de navigation_slider.js

Ici, le paramètre est **JTL=0,0**. Le choix du ou des paramètres est primordial pour avoir une image de petite taille. Au départ, les paramètres **HEI=250&QLT=90&CVT=jpeg** étaient utilisés. Ils avaient naïvement été récupérés d'une requête sans autre forme d'investigation (probablement de l'image dans la mini-map). En utilisant intelligemment ces paramètres, on peut diminuer significativement la quantité de données et gagner un temps de chargement considérable. En effet, l'image renvoyée par IIPImage avec les paramètres **HEI=250&QLT=90&CVT=jpeg** pèse 51 KB, alors que la même image avec les paramètres **JTL=0,0** ne pèse que 7 KB, soit une réduction de la taille d'environ 86%. Lorsqu'un cahier contient une centaine de pages, la différence se fait alors rapidement ressentir.

L'image est affichée comme une image normale, avec une hauteur ("height") définie. Cela est nécessaire pour que toutes les images d'un même cahier gardent des proportions comparables entre elles. Un événement "click" est associé à chaque image. Lors du clic, l'image est sélectionnée (son cadre devient jaune) et un événement "thumbnailClicked" est *fired*. Celui-ci est utilisé dans **mass_editor.js** pour changer l'image du viewer sans recharger toute la page. L'identifiant qui lui est passé en paramètre ("this.id") permet d'identifier l'image dans le DOM et ainsi de récupérer d'autres données (les attributs HTML5 "data-imageid" et "data-imagepath") qui ont été placées dans la balise "img" et qui sont nécessaires à l'affichage.

Intégration dans IIPMooViewer

L'intégration dans IIPMooViewer est discutable. En effet, si cela simplifie certaines opérations, il n'est pas indispensable que le slider soit intégré au viewer. Contrairement à la barre d'outils et au module d'annotation présentés plus tôt, le slider utilise un projet externe SlideItMoo : ce projet est loin d'être abouti et sa personnalisation n'est pas aisée. Il serait préférable d'écrire un module dédié pour IIPMooViewer, qui s'intégrerait mieux (comme celui de la *National Gallery* de Londres [22]).

6.7.4 Application client personnalisée

6.7.4.1 iipmooviewer-2.0.js

iipmooviewer-2.0.js

La source d'IIPMooViewer a été modifiée pour certaines actions, car il n'était pas possible de faire autrement. Ces modifications ont toutes trait à la gestion de l'édition des annotations décrite ci-dessus.

22. `http://cima.ng-london.org.uk/iip/gallery/`

Ajout d'un test dans la fonction "requestImages" :

```
// Create new annotations and attach the tooltip to them if it already exists
if( this.annotations ){
  this.createAnnotations();
  // MASSIMO : add test to avoid showing annotations on mouseOver
  // while editing and after an event reloaded the page
  if( this.annotationTip && (this.annotationBeingEdited == null ) ) {
    this.annotationTip.attach( this.canvas.getChildren('div.annotation') );
  }
}
```

Dans **rotate : function(r)**, ajout d'un test pour empêcher la rotation et afficher un message si une annotation est en cours d'édition ou si l'utilisateur est en train de créer (dessiner) l'annotation.

```
//MASSIMO check if an annotation is being edited
var tempBool = false;
for( var a in this.annotations ){
  if( this.annotations[a].edit == true) {
  tempBool = true;
  break;
  }
}

//Massimo
if ( this.isCreatingAnAnnotation == true || tempBool == true ) {
  var msg = SaussureLang['cantRotateEditing'];
  alert(msg);
  return;
}
```

Ajout du *fire* d'un événement dans **positionCanvas : function()** :

```
  positionCanvas: function(){
  ...
// Necessary to detect a zoom or reset while creating an annotation
// (after pressing the create annotation button, but before starting to draw the annotation).
this.fireEvent('positionCanvas');
```

Copie de la fonction **reflow()** dans **newReflow()**. Identique sauf que la fonction **newReflow()** ne termine pas par requestImages().

```
newReflow: function(){
  ...
    // Update images
    // this.requestImages();
```

6.7.4.2 concepts_saussuriens.js

Les "concepts" utilisés dans ce travail sont des concepts sémantiques tirés du lexique de la terminologie saussurienne réalisée par Nilda RUIMY et Silvia PICCINI. Ce dernier est disponible en ligne grâce à un navigateur d'ontologie [23].

Le fichier **concepts_saussuriens.js** déclare simplement un tableau associatif "cA" (pour "conceptArray") dont chaque concept est la clef, et le lien URL correspondant est la valeur (concept=>URL). Pour réduire la taille du fichier, le lien est construit par concaténation de trois *strings* : le début de l'URL (urlA), l'identifiant du concept (le *string*) et la fin de l'URL (urlB). En effet, seul l'identifiant change et cela permet de réduire considérablement la taille du fichier source :

```
var urlA = "http://www.ilc.cnr.it/lessico/browser/individuals/";
var urlB = "/?session=13ef5786fed-188-13ef57a75a6";
var cA={
```

23. http://www.ilc.cnr.it/viewpage.php/sez=ricerca/id=917/vers=ita

```
"ablaut":urlA+"-1061716271"+urlB,
"abstraction1":urlA+"-394506073"+urlB,
...
};
```

6.7.4.3 iip_semantic_web_brero.js

Le fichier **iip_semantic_web_brero.js** contient un ensemble de fonctions pour la communication avec le service frontal. On peut le considérer comme la version Javascript du script PHP **sparql_request_pool.php** décrit dans la section 6.4.

Le listing 6.21 présente les entêtes des fonctions du script **iip_semantic_web_brero.js**.

```
getXFromFilename(action, searchfor, imagesfilename, annotationArray, htmlElement)

// Wrappers autour "getXFromFilename"
getAnnotationsFromFilename(imagesfilename, annotationArray)
getTranscriptionsFromFilename(imagesfilename, annotationArray, htmlElement)
getTranscriptionsFromFilenameAsAnnotations(imagesfilename, annotationArray)
getConceptualAnnotationsFromFilename(imagesfilename, annotationArray)

getThumbnailsFromFilename(imagesfilename, dataThumb, DEBUG_MODE)

addNewAnnotConceptualAnnotOrTrans(action,addParameter,imagesfilename,newAnnot,iipMooViewerInstance)

// Wrappers autour de "addNewAnnotConceptualAnnotOrTrans"
addNewAnnotation(imagesfilename, newAnnot, iipMooViewerInstance)
addNewTranscriptionElement(imagesfilename, newAnnot, iipMooViewerInstance)
addNewConceptualAnnotation(imagesfilename, newAnnot, iipMooViewerInstance)

editOrDeleteAnnotationOrTranscription(action, annotOrTransParameter, annotOrTrans)

// Wrappers autour de "editOrDeleteAnnotationOrTranscription"
deleteAnnotation(newAnnot)
editAnnotation(newAnnot)
deleteConceptualAnnotation(newAnnot)
editConceptualAnnotation(newAnnot)
deleteTranscriptionElement(newAnnot)
editTranscriptionElement(newAnnot)
```

Listing 6.21 – Entêtes de fonction du script **iip_semantic_web_brero.js**

La fonction **getXFromFilename** permet, selon les paramètres qui lui sont fournis, de retourner des listes de résultats. Des *wrappers* permettent de simplifier la lisibilité du code : **getAnnotationsFromFilename** pour une liste d'annotations textuelles, **getTranscriptionsFromFilenameAsAnnotations** pour une liste d'éléments de transcription, **getConceptualAnnotationsFromFilename** pour une liste d'annotations conceptuelles et **getTranscriptionsFromFilename** pour la liste des transcriptions (celles affichées dans le panneau de droite de l'interface utilisateur).

La fonction **getThumbnailsFromFilename** retourne une liste de noms de fichiers utilisée pour la génération du slider de *thumbnails*.

La fonction **addNewAnnotConceptualAnnotOrTrans** ajoute du contenu dans le triplestore et retourne le résultat de l'opération. Des *wrappers* permettent de simplifier la lisibilité du code : **addNewAnnotation** pour ajouter une annotation textuelle, **addNewTranscriptionElement** pour ajouter un élément de transcription et **addNewConceptualAnnotation** pour ajouter une annotation conceptuelle. Ces fonctions retournent le résultat de la requête (échec ou succès), ainsi qu'un *string* en JSON permettant de mettre à jour les données locales de l'annotation, pour qu'elle reflète celle qui a été fraîchement ajoutée dans le triplestore. En effet, comme expliqué dans le chapitre 4, l'identifiant de l'annotation est généré sur le serveur. Si le client ne met pas à jour son identifiant local, l'utilisateur ne pourra, par exemple, pas éditer l'annotation après l'avoir sauvée.

La fonction **editOrDeleteAnnotationOrTranscription** permet de modifier ou de supprimer du contenu existant. Là encore, des *wrappers* permettent de simplifier la lisibilité du code : **deleteAnnotation** pour effacer une annotation, **editAnnotation** pour modifier une annotation, et ainsi de suite.

Tous les *wrappers* fonctionnent par passage de paramètres textuels :

```
// GET_ANNOTATIONS_FROM_FILENAME
function getAnnotationsFromFilename(imagesfilename, annotationArray) {
  getXFromFilename('QUERY', 'annotations', imagesfilename, annotationArray);
}
```

Les fonctions appelées par les *wrappers* font toutes des appels au service frontal grâce à un objet XMLHttpRequest. Elles construisent les requêtes HTTP avec les paramètres fournis, puis les envoient sur le service frontal.

Le listing 6.22 illustre un cas typique de construction d'une requête HTTP avec XMLHttpRequest. Comme le service frontal est sur le même serveur que le serveur web, il est suffisant d'utiliser le nom du script (**annotations.php**) plutôt que l'URL complète.

Comme la majorité des scripts Javascript sont asynchrones, on doit forcer ici l'exécution d'une requête synchrone (asyncFlag=false), sans quoi l'exécution du code (celui qui a appelé la fonction getXFromFilename) continuerait sans avoir encore reçu l'information du serveur. Le traitement de la réponse est effectué dans la fonction associée à **http.onreadystatechange**.

```
function getXFromFilename(action, searchfor, imagesfilename, annotationArray, htmlElement) {
  var http = new XMLHttpRequest();
  var url = "annotations.php";
  var params = "";
  params += "ACTION="+action+"&";
  params += "imagesfilename="+imagesfilename+"&";
  params += "GET_FROM_FILENAME="+"true"+"&";
  params += "SEARCHFOR="+searchfor+"&";
  asyncFlag = false; // "true = requete asynchrone"

  http.open("POST", url, asyncFlag);
  http.setRequestHeader("Content-type", "application/x-www-form-urlencoded ; charset=UTF-8");
  http.setRequestHeader("Accept", "application/sparql-results+json, */*;q=0.5");

  http.onreadystatechange = function() {//Call a function when the state changes.
    ...
  }
  http.send(params);
}
```

Listing 6.22 – Extrait de la fonction getXFromFilename

6.7.4.4 mass_editor.js

Le script **mass_editor.js** est celui qui instancie toute l'application client. Le corps (`body`) de la page générée par **viewer_visitor.php** en PHP ne contient presque rien. Elle charge les modules Javascript et les éléments (`div`) qui seront utilisés par l'application Javascript. Tout le reste est effectué dans **mass_editor.js**.

Lors du chargement de la page, les modules Javascript sont chargés (dans l'ordre de la liste de la section 6.7.2).

Les paramètres (GET/POST) déterminent le contenu (quel manuscrit afficher, etc.). Tout le contenu est récupéré grâce à des appels effectués par les fonctions du script **iip_semantic_web_brero.js**

Tout d'abord, la gestion de l'éditeur externe (TinyMCE) est déclarée en ajoutant les fonctions nécessaires avec "*implement*" dans la classe IIPMooViewer. C'est notamment ici

qu'est déclaré le contenu du formulaire (par exemple, la liste déroulante pour les annotations conceptuelles, générées depuis **concepts_saussuriens.js**), ou encore que l'on spécifie comment récupérer le contenu de l'éditeur externe dans **externalAnnotationEditorGetText**, etc.

```
Contient
IIPMooViewer.implement({
  externalAnnotationEditorSetHTML() { ...}
  externalAnnotationEditorCreate: function(){ ... }
  externalAnnotationEditorGetText: function(){
    ...
    tinymce.get('tinymce').getContent();
    ...
  }
  externalAnnotationEditorGetCategory: function(){ ... }
  externalAnnotationEditorGetTitle: function(id){ ... }
  externalAnnotationEditorSuppress: function(){ ... }
});
```

Une instance de la classe IIPMooViewer est créée :

```
var server = '/cgi-bin/iipsrv.fcgi';

// Create our iipmooviewer object
var iipmooviewer = new IIPMooViewer( "viewer", {
server: server,
image: image,
credit: credit
});
```

Des événements sont alors liés à l'instance d'IIPMooViewer :

```
// Triggered when a "delete" event occurs on an annotation
iipmooviewer.addEvent('annotationDelete', function(annotationToDelete){
  if ( annotationMode == "transcription" ) {
    deleteTranscriptionElement(annotationToDelete);
  } else if ( annotationMode == "conceptualannotation" ) {
    deleteConceptualAnnotation(annotationToDelete);
  } else {
    deleteAnnotation(annotationToDelete);
  }
});

iipmooviewer.addEvent('annotationAdd', function(annotationToAdd){ ... }

iipmooviewer.addEvent('annotationEdit', function(annotationToEdit){ ... }
```

Ces fonctions sont appelées lorsque l'événement associé est *fired*. Par exemple, la procédure d'effacement est exécutée, car le bouton *delete* du formulaire *fire* l'événement (*fireEvent*) "annotationDelete" (ce code est déclaré dans **annotations-edit-mass.js**) :

```
 _this.fireEvent('annotationDelete', _this.annotations[id]);
```

La détection de l'événement "thumbnailClicked" permet de savoir qu'un *thumbnail* a été cliqué dans le slider. L'identifiant lui est passé en paramètre et permet de récupérer les valeurs des attributs HTML5 (voir la section 6.7.3.4). Ces valeurs sont passées en paramètre à la fonction **changeCurrentImage** qui va changer l'image du viewer, recharger les annotations, etc.

```
iipmooviewer.addEvent('thumbnailClicked', function(imageId){
  var thumbnailClicked = $(imageId);
  var imagepath = thumbnailClicked.get('data-imagepath');
  var thumbImagename = thumbnailClicked.get('data-imageid');
  changeCurrentImage(this, imagepath, thumbImagename);
}

function changeCurrentImage(iipViewerInstance, newImagePath, newImageName){
...
  reloadImageAndData(iipViewerInstance, newImagePath, newImageName);
}
```

La fonction **reloadImageAndData** sert à mettre à jour l'information de la page. Elle est appelée à la suite d'un bon nombre d'événements : le clic sur un *thumbnail*, lorsque l'un des

boutons de changement de mode a été cliqué, une fois que le DOM est chargé (événement **domready**), etc.

```
function reloadImageAndData(iipViewerInstance, newImagePath, newImageName){ ... }
```

Plusieurs appels sont effectués dans les fonctions du script **iip_semantic_web_brero.js**.

Les fonctions associées aux boutons sous l'image permettent de changer de mode en modifiant une variable et en appelant **reloadImageAndData**.

```
function switchToAnnotation(me){
  ...
  annotationMode = "annotation";
  ...
  reloadImageAndData(iipmooviewer, imagespath, imagesfilename);
}
function switchToTranscription(me){
function switchToConceptualAnnotation(me){
```

La dernière fonction importante est celle liée à l'événement DOMready.

```
window.addEvent('domready',function() { ... }
```

Elle effectue la plupart des instanciations, notamment en appelant les fonctions décrites dans les différents fichiers présentés ci-dessus : **getAnnotationsFromFilename**, **getTranscriptionsFromFilename**, **getThumbnailsFromFilename**, **iipmooviewer.createThumb**, **iipmooviewer.createExtendedToolbar**, **reloadImageAndData**, etc.

Finalement **reloadImageAndData** est appelé et la page en enfin complètement chargée !

6.7.4.5 addTinyMCE.js

Le fichier **addTinyMCE.js** contient la déclaration de l'instanciation (avec les paramètres requis) de l'éditeur WYSIWYG TinyMCE.

```
tinymce.init({ ... })
```

Il contient également une fonction pour supprimer proprement l'éditeur :

```
function removeTinymceById(tinymceId)
```

6.8 Exemple d'une action détaillée

Dans cette section, on détaille une action qui illustre le comportement de l'application et qui montre comment les différentes parties fonctionnent entre elles. Il n'est pas possible de détailler toutes les fonctionnalités, mais celle présentée ici est représentative et permet de facilement l'appliquer aux autres. Seuls les détails techniques les plus importants sont décrits ici.

6.8.1 Ajout d'une annotation

Lorsque l'utilisateur visualise un manuscrit (**viewer_visitor.php**), il peut créer une annotation. Il doit être authentifié pour que le bouton de création soit cliquable.

Si l'utilisateur est dans le mode "Annotations textuelles", il clique sur le bouton "Créer une annotation". Lorsque la souris est sur l'image, l'icône du curseur change pour indiquer que l'on peut créer une zone. L'utilisateur clique et laisse le bouton enfoncé, déplace la souris pour

dessiner la zone et relâche le clic pour terminer la création de la zone. A ce moment, le viewer et la zone de transcription à droite sont redimensionnés (réduits) pour faire apparaître un éditeur WYSIWYG. L'utilisateur saisit le texte souhaité ; il peut également encore redimensionner la zone de l'annotation. Lorsqu'il est satisfait, il clique sur *ok* : l'annotation est sauvée, l'éditeur disparaît de façon à ce que le viewer et la zone de transcription reprennent leur taille initiale.

Reprenons ces étapes à partir du clic pour créer l'annotation.

Lors du clic, la coordonnée de départ (x,y) est détectée et sauvée temporairement. Il s'agit d'une coordonnée relative (*float* entre 0 et 1). En effet, il est indispensable d'utiliser un tel système de coordonnées pour que la zone puisse être reproduite "au bon endroit" à n'importe quel niveau de zoom. À chaque déplacement de la souris (*mouseMove*), la coordonnée actuelle est détectée. Cela permet de calculer la hauteur relative et la largeur relative de la zone par rapport au point de départ, et de dessiner la zone au fur et à mesure que la souris bouge. Lorsque le clic est relâché, la dernière position de la souris est sauvée et une zone temporaire est créée sur l'image.

Les propriétés de style (CSS) sont manipulées avec du JavaScript pour redimensionner les zones du viewer et des transcriptions. Un éditeur WYSIWYG (TinyMCE) est initialisé dans l'espace récupéré. La saisie du texte (HTML) est gérée par TinyMCE. Lors de la sauvegarde du texte HTML, les coordonnées relatives (x,y,h,w) et la catégorie sont encodés en JSON et envoyés sur le service frontal (**annotations.php**). Les données sont "*URL encoded*" :

Envoi sur : http://129.194.186.2/iipmooviewer/annotations.php
Données :

```
ACTION:ADD
imagesfilename:ms_fr_03951_10_f028v_029.jp2
newAnnot:{"x":0.11674008810572688,"y":0.07462686567164178,
"w":0.19383259911894274,"h":0.05970149253731343,"id":"hkmm2opx",
"category":"annotationCategory","title":"",
"text":"<p>Ceci est une annotation</p>"}
```

Lorsque le script **annotation.php** reçoit ces données, il vérifie la présence du paramètre "ACTION" (voir le listing 6.23). Si la valeur de celui-ci est égale à "ADD", il vérifie ensuite la présence des autres paramètres "imagesfilename" et "newAnnot". Si ceux-ci existent, il récupère le nom du fichier image ("imagesfilename") et convertit l'annotation ("newAnnot") de JSON vers un tableau.
N.B. : *The superglobals $_GET and $_REQUEST are already decoded. Using urldecode() on an element in $_GET or $_REQUEST could have unexpected and dangerous results.* [24]

```php
if (array_key_exists('newAnnot',$_POST) && array_key_exists('imagesfilename',$_POST)){
   $imagesfilename = $_REQUEST["imagesfilename"];
   $resultNewAnnotArray = json_decode($_REQUEST["newAnnot"], true);

   $resultArray = addNewAnnotation($imagesfilename, $resultNewAnnotArray, $prefixes);

   $resultInsert = $resultArray[0];
   $resultQuery = $resultArray[1];
   $jsonReturned = $resultArray[2];
   printAddNewAnnotation($resultInsert, $resultQuery, $jsonReturned);
}
```

Listing 6.23 – Algorithme principal (test des paramètres) d'**annotations.php**

24. http://php.net/manual/en/function.urldecode.php

Ceux deux paramètres et les préfixes SPARQL sont passés à la fonction "**addNewAnnotation**" du fichier **sparql__request__pool.php** (*backend*). La fonction **addNewAnnotation** (listing 6.24) est un simple *wrapper* autour de la fonction **addNewX**. Les paramètres sont transmis tels quels à **addNewX**, ainsi que trois *strings* spécifiques à l'ajout d'une annotation (ils sont différents s'il s'agit d'une annotation conceptuelle ou d'un élément de transcription).

La fonction **addNewX** contient toute l'intelligence et envoie plusieurs requêtes au triplestore. Elle est détaillée plus tard dans cette section.

```
function addNewAnnotation($imagesfilename, $resultNewAnnotArray) {
   return addNewX($imagesfilename, $resultNewAnnotArray,
           "_annot_001", "hasAnnotation", "Annotation");
}
```

Listing 6.24 – Fonction addNewAnnotation() de **sparql__request__pool.php**

Une requête est construite comme un *string*, puis elle est exécutée avec la fonction **sparql__curl__request** de **sparql__php__curl__lib.php**. La fonction **sparql__curl__request** (listing 6.25) est un *wrapper* autour de la fonction **sparql__curl__query**. Celle-ci exécute la requête en l'envoyant à l'adresse du *endpoint* avec cURL. Toutes les requêtes, qu'elles soient de recherche ou d'ajout, fonctionnent sur ce même principe. Seuls quelques paramètres et l'adresse du *endpoint* changent. La requête d'ajout retourne les résultats en JSON.

```
function sparql_curl_query($endpoint, $sparql_query) {
   $ch= curl_init(); // get curl handle
   $query_string = 'query='.urlencode($sparql_query).'&';
   $opt_array = array(
      CURLOPT_URL => $endpoint,
      CURLOPT_POST => TRUE,
      CURLOPT_POSTFIELDS => $query_string ,
      CURLOPT_HTTPHEADER => array("Accept: application/sparql-results+json, */*;q=0.5",
           "Content-Type: application/x-www-form-urlencoded ; charset=UTF-8"),
      CURLOPT_RETURNTRANSFER => true
   );
   curl_setopt_array($ch, $opt_array); //Set the array of options
   $response = curl_exec($ch); // Execute
   curl_close($ch); // close
   return $response;
}
```

Listing 6.25 – Fonction sparql_curl_query de **sparql__php__curl__lib.php**

La première étape de la fonction **addNewX** (listing 6.26) consiste à effectuer un contrôle pour vérifier que l'utilisateur est authentifié, grâce à la fonction **isUserLoggedIn()** de UserCake. Si tel est le cas, alors différentes variables sont sauvées : la date courante, le nom de l'utilisateur authentifié, ainsi que l'URI utilisé pour le type RDF **saussure:Photo** (construit depuis le nom de l'image).

Une première requête est construite. Elle permet de déterminer le nombre de zones qui sont liées à l'image (propriété **p:contient**). Ce nombre est converti en string et permet de construire le suffixe à concaténer au nom de l'image pour générer l'URI de la zone. Par exemple pour l'image "ms_fr_03951_10_f028v_029", la zone sera "ms_fr_03951_10_f028v_029**__Z__001**". La présence de l'URI est vérifiée dans le triplestore avec une requête ASK. Si celle-ci retourne "*false*", cela signifie que l'URI est bel et bien unique. Par contre, si une zone avec cette URI existe, la valeur numérique du préfixe est incrémentée ("**__Z__001**" devient "**__Z__002**", etc.) et la requête ASK est répétée jusqu'à ce qu'elle retourne "*true*".

ATTENTION : Cette technique permet de trouver une URI non utilisée, mais pas unique. Cela devra impérativement être modifié dans le *workflow* et la convention de nommage des URI. Il faut à la place créer une fonction de génération de suffixe unique : il peut s'agir du même principe que ci-dessus, sauf qu'à chaque génération d'une URI, une référence à celle-ci est sauvée dans le triplestore. Il suffirait alors d'effectuer un SELECT sur cette valeur lorsqu'on souhaite une nouvelle zone, puis de l'incrémenter pour obtenir un identifiant unique. La référence devrait alors être mise à jour avec la nouvelle valeur générée (attention, ces opérations devraient être atomiques).

À ce moment, selon le modèle établi et la convention de nommage des URI, on part du principe que la forme et l'annotation n'existent pas. Cette URI sert alors à la génération de celles de la forme et de l'annotation : "ms_fr_03951_10_f028v_029_Z_001_Shape_001" et "ms_fr_03951_10_f028v_029_Z_001_annot_001".

La requête d'ajout peut être générée avec ces différentes URI, et les paramètres fournis (texte, coordonnées de la zone, etc.), ainsi que le nom de l'utilisateur, l'heure et la date d'ajout. Le texte est encore converti en texte brut avec la fonction **html2TextRegexExAndStrip**. Les deux versions du texte sont insérées (propriétés **p:texte** et **p:rawText**).

La requête est exécutée (voir le listing 6.28 pour la requête générée). Son retour ainsi que les URI (utilisées pour cet ajout) sont retournés.

```
function addNewX($imagesfilename, $resultNewAnnotArray, $xSuffix, $xProperty, $xType){
   if( isUserLoggedIn() ) { # BEGIN AUTHENTICATION CHECK
      global $prefixes;
      global $evaluationQueriesEndpoint;
      global $modificationQueriesEndpoint;

      global $loggedInUser;
      $usernameTemp = $loggedInUser->username ;
      $currentDate = date("Y-m-d_H-i-s");

      $imagenameDOT = str_replace( '.' , '-DOT-', $imagesfilename);

      $sparql_mode = "query";
      $graph_file = "<file://test_001_2013-04-08.xml>";

      $query =
         $prefixes . "
         SELECT (count(?zone) as ?c)
         FROM " . $graph_file . "
         WHERE {
            saussure:$imagenameDOT p:contient ?zone .
         }
      ";
      $resultCount=sparql_curl_request($sparql_mode,$evaluationQueriesEndpoint,$query);

      // Decodes the variable from received JSON
      $resultCountArray = json_decode($resultCount, true);
      $resultCountValue = $resultCountArray["results"]["bindings"][0]["c"]["value"];

      // Checks if exists <=> tant que ASK retourne vrai (= le triplet existe)
      $tempZoneName = "";
      $resultAsk = 'true';
      while ( strcmp($resultAsk,"true") == 0) { // si vrai = 0
         $resultCountValue++;
         // convert number to format : 3 => 003 / 14 => 014 / 500 => 500
         $formattedCountValue = sprintf('%03d', $resultCountValue);
         $suffix = "_Z_" . $formattedCountValue ; // . "_annot_001";
         // Replaces extension - preg_replace(needle, replacement, stack)
         $tempZoneName = preg_replace( '/\..*$/' , $suffix , $imagesfilename);
         $query =
            $prefixes . "
            ASK {
               saussure:$imagenameDOT p:contient saussure:$tempZoneName .
               saussure:$tempZoneName rdf:type saussure:Zone .
            }
         ";
         $resultAsk=sparql_curl_request($sparql_mode,$evaluationQueriesEndpoint,$query);
      }

      // Comme la zone n'existait pas, on part du principe que l'annotation non plus
      $tempAnnotName = $tempZoneName . $xSuffix;
      $tempShapeName = $tempZoneName ."_Shape_001";


      // INSERT NEW ZONE + SHAPE + ANNOTATION
      $sparql_mode = "update";
      $query = $prefixes .'
```

```
      INSERT DATA {
         GRAPH '. $graph_file .' {
            # INSERT ZONE dans Photo
            saussure:'. $imagenameDOT .' rdf:type saussure:Photo ;
            p:contient saussure:'. $tempZoneName .' .
            # CREATE ZONE + annotation + shape
            saussure:'. $tempZoneName .' rdf:type saussure:Zone ;
               p:'.$xProperty.' saussure:' .$tempAnnotName .' ;
               p:hasForme saussure:'. $tempShapeName .' .

            saussure:'. $tempAnnotName .' rdf:type saussure:'.$xType.' ;
               p:texte """'. str_replace("\"", "\\\"",
                   str_replace("\\", "\\\\", $resultNewAnnotArray["text"])) .'""" ;
               p:rawText """'. str_replace("\"", "\\\"",
                 str_replace("\\", "\\\\",
                                html2TextRegexExAndStrip($resultNewAnnotArray["text"])
                 )) .'""" ;
               p:hasCreatorName """'. $usernameTemp .'""" ;
               p:hasCreationDate """'. $currentDate .'""" ;
               p:hasEditorName """'. $usernameTemp .'""" ;
               p:hasEditionDate """'. $currentDate .'""" .

            saussure:'. $tempShapeName .' rdf:type saussure:FormeGeometrique ;
               p:hasX "'. $resultNewAnnotArray["x"] .'";
               p:hasY "'. $resultNewAnnotArray["y"] .'";
               p:hasHeigth "'. $resultNewAnnotArray["h"] .'";
               p:hasWidth "'. $resultNewAnnotArray["w"] .'" .
         }
      }
      ';

      $resultInsert=sparql_curl_request($sparql_mode,
                           $modificationQueriesEndpoint,$query);

      $jsonToReturn = "{";
      $jsonToReturn .= "Forme:'http://saussure.com/ressource/$tempShapeName',";
      $jsonToReturn .= "SurfaceCouverte:'http://saussure.com/ressource/$imagenameDOT',";
      $jsonToReturn .= "Zone:'http://saussure.com/ressource/$tempZoneName',";
      $jsonToReturn .= "id:'http://saussure.com/ressource/$tempAnnotName'";
      $jsonToReturn .= "}";

      // Result, query, and annotationId to replace in the client DOM
      return array($resultInsert, $query, $jsonToReturn);
   } else { # END AUTHENTICATION CHECK
      return array("NOT_AUTHENTICATED-NOTHING_WAS_DONE",
           "NOT_AUTHENTICATED-NOTHING_WAS_DONE","NOT_AUTHENTICATED-NOTHING_WAS_DONE");
   }
}
```

Listing 6.26 – Fonction addNewX() du fichier **sparql_request_pool.php**

On revient à l'exécution du test du script **annotations.php** (voir le listing 6.23). Le retour de **addNewAnnotation** est un tableau. Chaque élément est sauvé dans une variable pour plus de lisibilité, puis ceux-ci sont passés en paramètre à la fonction **printAddNewAnnotation** du fichier **display_sparql_lib.php** (listing 6.28).
N.B. : Le terme "print" est utilisé par abus de langage, car la fonction utilise des "echo". Ces résultats sont en réalité retournés au client dans l'application Javascript, ils ne sont pas réellement affichés dans une page web.

```php
function printAddNewAnnotation($resultInsert, $query, $jsonToReturn) {
   if( !isset($resultInsert) || preg_match("/^MALFORMED QUERY/i", $resultInsert) ) {
      echo "ERROR ! Annotation ADD, returned : ";
      echo "JSONTORETURN:$jsonToReturn";
      echo "\n$resultInsert\n$query\n";
      echo "";
   } else {
      echo "Annotation added successfully : \n";
      echo "JSONTORETURN:$jsonToReturn";
      echo "\n$resultInsert\n$query\n";
      echo "";
   }
}
```

Listing 6.27 – Fonction printAddNewAnnotation() de **display_sparql_lib.php**

Le client reçoit en retour un message (succès/échec) et des données formatées en JSON. Celles-ci lui permettent de mettre à jour les données locales (identifiant de l'annotation, etc.). Un exemple de message de succès avec les données en JSON :

```
Annotation added successfully :
JSONTORETURN:{
 Forme:'http://saussure.com/ressource/ms_fr_03951_10_f028v_029_Z_001_Shape_001',
 SurfaceCouverte:'http://saussure.com/ressource/ms_fr_03951_10_f028v_029-DOT-jp2',
 Zone:'http://saussure.com/ressource/ms_fr_03951_10_f028v_029_Z_001',
 id:'http://saussure.com/ressource/ms_fr_03951_10_f028v_029_Z_001_annot_001'}
```

La requête SPARQL utilisée pour l'ajout est également retournée à titre informatif. Elle est mise en forme dans le listing 6.28.

```
PREFIX rdf:<http://www.w3.org/1999/02/22-rdf-syntax-ns#>
PREFIX rdfs:<http://www.w3.org/2000/01/rdf-schema#>
PREFIX p:<http://saussure.com/property/>
PREFIX saussure:<http://saussure.com/ressource/>

INSERT DATA {
   GRAPH <file://test_001_2013-04-08.xml> {
      # INSERT ZONE dans Photo
      saussure:ms_fr_03951_10_f028v_029-DOT-jp2 rdf:type saussure:Photo ;
      p:contient saussure:ms_fr_03951_10_f028v_029_Z_001 .
      # CREATE ZONE + annotation + shape
      saussure:ms_fr_03951_10_f028v_029_Z_001 rdf:type saussure:Zone ;
         p:hasAnnotation saussure:ms_fr_03951_10_f028v_029_Z_001_annot_001 ;
         p:hasForme saussure:ms_fr_03951_10_f028v_029_Z_001_Shape_001 .

      saussure:ms_fr_03951_10_f028v_029_Z_001_annot_001 rdf:type saussure:Annotation ;
         p:texte """<p>Ceci est une annotation</p>""" ;
         p:rawText """Ceci est une annotation""" ;
         p:hasCreatorName   """massimo""" ;
         p:hasCreationDate   """2013-07-21_16-04-04""" ;
         p:hasEditorName   """massimo""" ;
         p:hasEditionDate   """2013-07-21_16-04-04""" .

      saussure:ms_fr_03951_10_f028v_029_Z_001_Shape_001 rdf:type saussure:FormeGeometrique ;
         p:hasX "0.11674008810573";
         p:hasY "0.074626865671642";
         p:hasHeigth "0.059701492537313";
         p:hasWidth "0.19383259911894" .
   }
}
```

Listing 6.28 – Requête SPARQL générée par **addNewAnnotation**

Chapitre 7

Test d'utilisabilité

Cette section décrit la manière dont les tests d'utilisabilité ont été menés, la méthode utilisée, les conditions d'évaluation, le matériel utilisé et les résultats obtenus.

On cherche à tester l'utilisabilité, c'est-à-dire à vérifier que le système fonctionne à la satisfaction des utilisateurs.

7.1 Méthodologie

7.1.1 Nombre de participants

Le nombre de participants requis dépend de ce que l'on veut réellement évaluer.

7.1.1.1 Recherche d'erreurs ou de problèmes : 5 utilisateurs

Selon Jakob Nielsen [1], la majorité des tests d'utilisabilité sont des études qualitatives [2], et une étude qualitative ne nécessite que cinq participants [3].

Il affirme que c'est avec les cinq premiers participants d'une étude que la majorité (85%) des problèmes d'utilisabilité sont détectés. En effet, on remarque que ces cinq participants font parfois des erreurs similaires ou butent parfois sur les mêmes tâches. Même si chacun réussit ou échoue l'exécution d'une partie des tâches, on observe généralement un chevauchement des erreurs commises par l'ensemble des utilisateurs.

Avec cinq participants, les faiblesses principales du système sont d'ores et déjà mises en avant. Si l'on continue l'évaluation, parmi les participants supplémentaires, certains vont également buter sur les mêmes tâches et faire les mêmes erreurs que les précédents. Évidemment, certains vont aussi mettre en avant de nouveaux problèmes d'utilisabilité, mais à ce stade, il n'est plus rentable de passer du temps à observer les mêmes problèmes se répéter. Il est alors recommandé de faire trois séries de tests avec cinq utilisateurs (3x5).

La première série permet de détecter 85% des problèmes d'utilisabilité. Des modifications sont faites pour corriger ces problèmes.

1. Jakob Nielsen est un expert dans le domaine de l'utilisabilité : `http://www.nngroup.com/people/jakob-nielsen/`
2. `http://www.nngroup.com/articles/how-many-test-users/`
3. `http://www.nngroup.com/articles/why-you-only-need-to-test-with-5-users/`

La deuxième série permet de vérifier que les modifications apportées ont bien amélioré les résultats, et si c'est le cas, de détecter les 15% de problèmes restants. De nouvelles modifications sont faites pour corriger ces problèmes.

La troisième série permet de vérifier que les modifications apportées ont bien amélioré les résultats du deuxième test.

Le désavantage de cette méthode est qu'elle doit être organisée sur le moyen/long terme, avec des pauses suffisamment longues entre les sessions pour permettre la correction des bugs, voire la refonte du *design* de l'interface dans les cas extrêmes.

7.1.1.2 Preuve d'utilisabilité : autant d'utilisateurs que possible

Outre la recherche d'erreurs, l'évaluation du système dans ce travail porte également sur l'ergonomie de l'interface et son utilisabilité. Contrairement à l'évaluation en termes de recherche d'erreurs ou de problèmes d'interface, la preuve d'utilisabilité ne peut pas se limiter à seulement 5 personnes. En effet, plus l'échantillon est grand, plus il est facile de tirer des conclusions sur les résultats, ou de faire ressortir des groupes d'utilisateurs (par ex. : "est-ce que les moins de 30 ans réussissent mieux ?", "est-ce que les linguistes ont plus de facilité ?" ...).

7.1.1.3 Les participants

Quatorze volontaires ont participé à l'évaluation. Comme le nombre de participants était limité et qu'il n'était pas possible d'organiser plusieurs sessions pour la détection d'erreur (voir 7.1.1.1), un compromis a été trouvé.

Les deux premiers utilisateurs ont permis de noter les premières faiblesses du système (mode "recherche d'erreur"). L'interface a été très légèrement modifiée, et deux tâches ont été ajoutées aux scénarios. Pour les douze volontaires suivants, ni les scénarios ni l'interface n'ont été modifiés par la suite.

Compte tenu du fait que les modifications apportées au système étaient mineures, les deux premiers utilisateurs sont comptés dans l'analyse des résultats. Tous les participants ont donné leur consentement pour la publication anonyme de données récoltées durant l'évaluation. L'échantillon se constitue de 14 personnes, d'âge, de sexe et de formation différents. Le tableau 7.1 et la liste suivante synthétisent les données sociodémographiques récoltées :

- 14 participants, dont 10 femmes et 4 hommes, entre 24 et 60 ans.
- Avec des notions de linguistique : 7.
- Avec des connaissances de la linguistique selon Ferdinand de Saussure : 7.
- Expert de Ferdinand de Saussure : 2.
- Avec un rapport direct au manuscrit : 5.
- Formation/métier :
  - Linguistes : 7.
  - Archivistes/bibliothécaires/conservateurs : 3.
  - Historiens : 2.
  - Ingénieurs (informaticiens) : 2.
- Presque tous les participants (11) étaient de formation universitaire en sciences humaines (Master/Licence, Doctorat), sauf 3 (respectivement Master en Sciences, Bachelor en Sciences et CFC).

	U01	U02	U03	U04	U05	U06	U07	U08	U09	U10	U11	U12	U13	U14
1. Je suis à l'aise avec un ordinateur.	5	3	4	4	5	5	4	3	2	5	5	4	5	3
2. J'utilise au moins une fois par semaine un ordinateur dans le cadre de mon travail ou dans ma vie privée.	5	5	5	5	5	5	5	5	5	5	5	5	5	5
3. Je sais utiliser le navigateur web *Google Chrome* ou *Mozilla Firefox* .	5	5	5	5	5	5	5	4	5	5	5	5	5	5
4. J'ai des bonnes connaissances en linguistique.	1	5	1	2	1	5	2	5	5	5	2	1	5	5
5. J'ai étudié ou je connais la linguistique selon Ferdinand de Saussure.	1	5	1	1	1	3	1	5	3	5	1	2	5	4
6. Je suis un spécialiste de Ferdinand de Saussure.	1	5	1	1	1	1	1	5	1	2	1	1	3	3
7. J'ai déjà consulté ou étudié des manuscrits	1	5	3	5	1	1	5	5	5	3	5	5	1	1
8. J'étudie ou je travaille régulièrement avec des manuscrits.	1	5	3	5	1	1	3	3	5	2	5	5	1	1
9. Je suis au courant de ce projet et je l'ai déjà utilisé ou vu en action.	3	4	1	1	2	5	1	5	1	1	1	1	1	1
10. (Pour les hommes) J'aimerais avoir une moustache comme Ferdinand de Saussure.	5	1	1	5	5	5	5	1	-	-	2	-	-	1
Sexe	M	M	F	F	M	F	F	F	F	F	M	F	F	M
Age	25	-	30	35	24	-	35	60	53	31	39	59	37	51
Operating system : Windows (W), Ubuntu (U), Mac OS (M)	W	-	W	W	U/W	W	M	W	W	M	W	W	M/W	W
Navigateur web : Firefox (FF), Chrome (C), Internet Explorer (IE), Safari (S)	C	-	IE	FF	C/FF	FF	S/FF	FF	FF	S	F	F	C/FF	C/FF

FIGURE 7.1 – *Données sociodémographiques. Légende : 1=Pas du tout d'accord, 5=Tout à fait d'accord*

7.1.2 Méthode d'évaluation

7.1.2.1 Conditions d'évaluation

Les tests d'utilisabilité ont été effectués dans des lieux différents, mais dans des conditions similaires pour tous les participants [4] :

- Environnement calme, sans bruit parasite (musique, conversation, travaux...).
- Test sur le même ordinateur (Windows 7), branché à un écran externe de 19" (4:3), avec la même version de Google Chrome et la même souris.
- Le cache, l'historique et les cookies de Chrome ont toujours été supprimés en début de test.
- Aucune modification n'a été apportée au site après le début des tests (sauf ceux mentionnés plus haut).
- La connexion internet sans fil présente sur le lieu du test a toujours été utilisée.

Pour éviter qu'une notion d'apprentissage ne risque de perturber les résultats, les scénarios ont été effectués dans des ordres différents (voir tableau 7.2). Par contre, l'ordre des tâches d'un scénario est fixe.

L'évaluation se fait en trois temps :
- Observation des actions de l'utilisateur durant l'exécution des scénarios
- Discussion avec le participant en fin de test pour récolter ses remarques et propositions
- Analyse du formulaire SUS (System Usability Scale)

4. En annexe (Annexe E.1) se trouvent une photo du matériel d'évaluation, ainsi que la liste du matériel utilisé, d'une part par les participants, et d'autre part pour le serveur.

Ordre d'exécution des scénarios

U01	U02	U03	U04	U05	U06	U07	U08	U09	U10	U11	U12	U13	U14
1	1	1	2	3	2	4	2	3	4	1	4	3	3
2	2	2	3	2	4	3	1	4	2	4	2	2	1
3	3	3	1	4	1	2	3	2	1	2	3	1	2
4	4	4	4	1	3	1	4	1	3	3	1	4	4

FIGURE 7.2 – *Tableau des ordres d'exécution des scénarios*

7.1.2.2 Protocole de test

Le protocole de test complet est disponible en annexe (Annexe E.2). Il comprend :
- le détail du déroulement du test,
- le consentement de participation,
- le formulaire sociodémographique et le formulaire de compétences,
- le formulaire SUS (System Usability Scale),
- les quatre scénarios.

Les scénarios ont été écrits de façon à couvrir la majorité des fonctionnalités de l'interface, afin de pouvoir évaluer le système dans son ensemble. Ils se concentrent sur des tâches différentes, et les tâches d'un scénario s'enchaînent de façon à minimiser le temps de passage des utilisateurs. Même si l'objectif visé est similaire (par ex. : "Créer une transcription"), toutes les tâches sont différentes et contiennent des nuances.

Résumé des scénarios et des tâches

- SCÉNARIO 1
  - (S1-T1) Trouver un manuscrit donné par sa cote (recherche textuelle).
  - (S1-T2) Créer une nouvelle annotation textuelle.
- SCÉNARIO 2
  - Trouver un manuscrit donné grâce à son emplacement physique à la bibliothèque (section, boîte...).
    - (S2-T1.2) Trouver un manuscrit (1)
    - (S2-T1.5) Trouver un manuscrit (2)
  - (S2-T2) Lier un concept au manuscrit.
  - (S2-T3) Changer de manuscrit lors de la visualisation d'un manuscrit en utilisant la navigation par miniatures (thumbnails).
  - (S2-T4) En visualisant un manuscrit donné, supprimer une annotation existante.
  - (S2-T5) Créer une nouvelle annotation textuelle (texte mis en forme).
- SCÉNARIO 3
  - (S3-T1) Trouver la liste des manuscrits liés à un concept sémantique.
  - (S3-T2) Créer une nouvelle transcription.
  - (S3-T3) Créer une nouvelle transcription.
- SCÉNARIO 4
  - (S4-T1) Trouver un manuscrit via une recherche textuelle (mots-clefs) sur les transcriptions et les annotations existantes.
  - (S4-T2) Modifier une transcription existante.

7.2 Résultats des évaluations

Cette section décrit les résultats des évaluations, leurs interprétations et les conclusions qu'elles permettent de tirer.

Les critères de l'utilisabilité sont définis par :
- l'efficacité : le produit permet à ses utilisateurs d'atteindre le résultat prévu ;
- l'efficience : le résultat est atteint avec un effort moindre ou requiert un temps minimal ;
- la satisfaction : confort et évaluation subjective de l'interaction par l'utilisateur.

7.2.1 Efficacité

L'efficacité est la capacité à atteindre le résultat prévu (sans tenir compte ni du temps ni de l'effort nécessaire).

On souhaite savoir combien d'utilisateurs ont réussi l'exécution des tâches sans intervention de l'évaluateur et ceci dans un temps "raisonnable". La réalisation d'une tâche est relativement courte, mais l'utilisateur doit lire le scénario et comprendre ce qui lui est demandé. Le temps "raisonnable" maximum est fixé arbitrairement à 3 minutes. S'il est dépassé, la tâche est considérée comme ratée.

Le tableau de la figure 7.3 présente les résultats de réussite des tâches. On distingue trois cas :
- Réussi = 1
- Réussi avec intervention = 0.5
- Raté = 0

Identifiant de la tâche	U01	U02	U03	U04	U05	U06	U07	U08	U09	U10	U11	U12	U13	U14	Taux réussite par tâche
Scénario 1 - tâche 1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	100%
Scénario 1 - tâche 2	1	1	1	1	1	1	1	1	1	1	1	1	1	1	100%
Scénario 2 - tâche 1.2	1	1	1	1	1	1	1	1	1	1	1	1	1	1	100%
Scénario 2 - tâche 1.5	1	1	1	1	1	1	1	1	1	1	1	1	1	1	100%
Scénario 2 - tâche 2	1	1	1	1	1	1	0.5	0	1	1	1	1	1	1	88%
Scénario 2 - tâche 3	1	1	1	1	1	1	1	1	1	1	1	1	1	1	100%
Scénario 2 - tâche 4	1	1	1	1	1	1	0	0	1	1	1	0	1	1	75%
Scénario 2 - tâche 5	-	-	1	1	1	1	1	1	1	1	1	1	1	1	100%
Scénario 3 - tâche 1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	100%
Scénario 3 - tâche 2	1	1	1	1	1	1	1	1	1	1	1	1	1	1	100%
Scénario 3 - tâche 3	-	-	1	1	1	1	1	1	1	1	1	1	1	1	100%
Scénario 4 - tâche 1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	100%
Scénario 4 - tâche 2	1	0.5	1	1	1	1	1	1	1	1	1	0.5	1	1	96%
Taux réussite par utilisateur	100%	95%	100%	100%	100%	100%	88%	85%	100%	100%	100%	88%	100%	100%	

(Réussi = 1 ; réussi avec intervention = 0.5 ; raté = 0)

FIGURE 7.3 – *Tableau de données pour l'efficacité*

Le taux de succès global (moyenne sur toutes les tâches et pour tous les utilisateurs confondus) est de **97%**. L'efficacité du système est donc très bonne.

Les graphiques des figures 7.4 et 7.5 montrent un taux de succès de 100% pour respectivement 10 des 13 tâches, et 10 des 14 utilisateurs. Cela signifie que 4 utilisateurs ont échoué (ou buté sur) l'une des 3 tâches problématiques :

- (S2-T2) Lier un concept au manuscrit.
  - U07 : Problème sur la création de la zone (relâcher la souris hors de l'image). Intervention (0.5).
  - U08 : N'était pas dans le mode "Annotations conceptuelles". Intervention et dépassement du temps "raisonnable" (0).
    N.B. : Il s'agissait de la première tâche de création d'une zone pour l'utilisateur.
- (S2-T4) En visualisant un manuscrit donné, supprimer une annotation existante.
  - U07 : A effacé le texte de l'annotation, mais n'a pas utilisé le bouton *Delete* (0).
  - U08 : Cheminement correct, mais a fait *Cancel* au lieu de *Delete* (0).
  - U12 : A effacé le texte de l'annotation, mais n'a pas utilisé le bouton *Delete* (0).
- (S4-T2) Modifier une transcription existante.
  - U02 : Double-clic sur le texte (*tooltip*), et non sur la zone de l'image. Intervention (0.5).
  - U12 : Double-clic sur le texte (*tooltip*), et non sur la zone de l'image. Intervention (0.5).

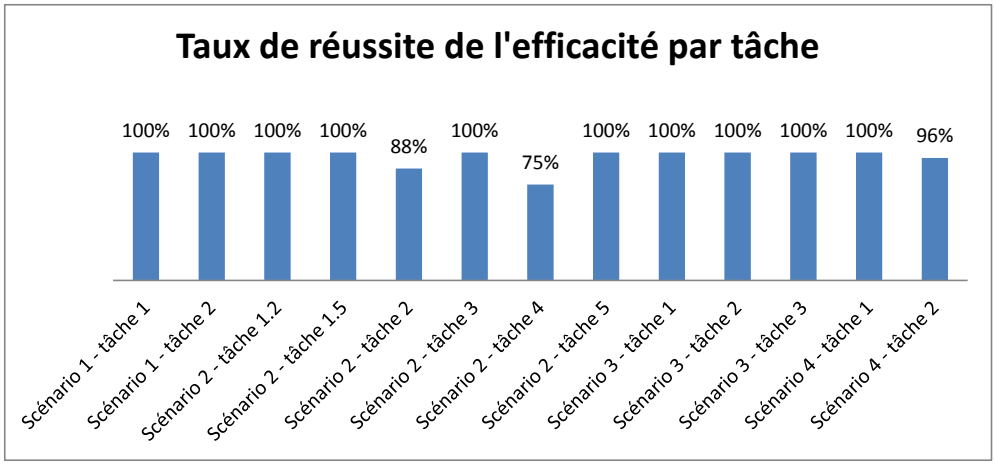


FIGURE 7.4 – *Taux de réussite de l'efficacité par tâche*

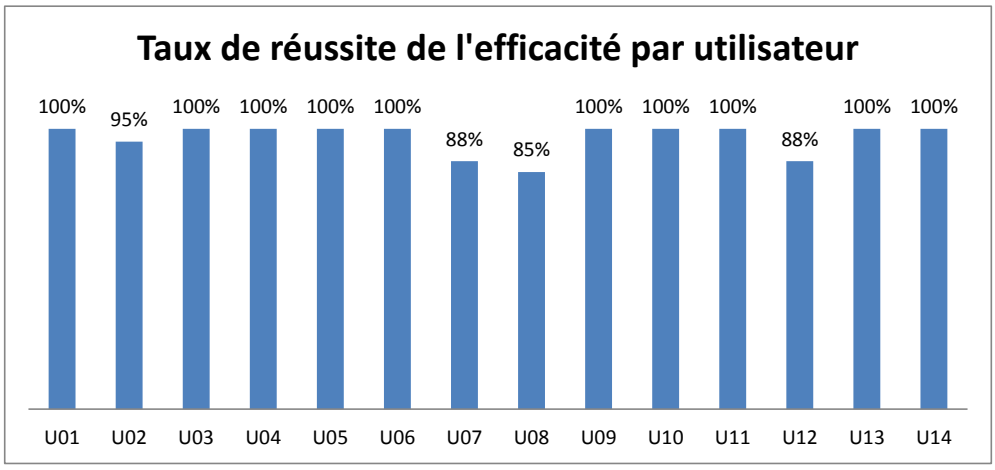


FIGURE 7.5 – *Taux de réussite de l'efficacité par utilisateur*

On comprend qu'il y a un réel problème au niveau de la suppression d'une annotation. La première personne a cliqué sur *cancel* par inattention, les deux suivantes ont effacé le contenu textuel, plutôt que de supprimer réellement l'annotation. Pour éviter cette confusion, le système pourrait avertir l'utilisateur au moment de la sauvegarde (*ok*) qu'il s'apprête à enregistrer une annotation avec un contenu vide, et lui proposer de l'effacer à la place. Dans tous les cas, il serait préférable d'interdire la sauvegarde d'annotations vides.

Le deuxième problème que l'on remarque concerne l'édition d'une transcription. En effet, plusieurs utilisateurs ont tenté de cliquer soit sur la zone d'affichage de la transcription (panneau de droite), soit sur le *tooltip*. D'autres utilisateurs l'ont également fait durant les tests, mais ils ont trouvé la solution par eux-mêmes. Ce problème est dû au fait que le contenu de la transcription est un texte, et que lorsqu'il est affiché dans le *tooltip*, ce dernier cache une bonne partie de la zone. De plus, la zone du *tooltip* occupe une plus grande surface que celle de la zone de l'image (voir 7.6).

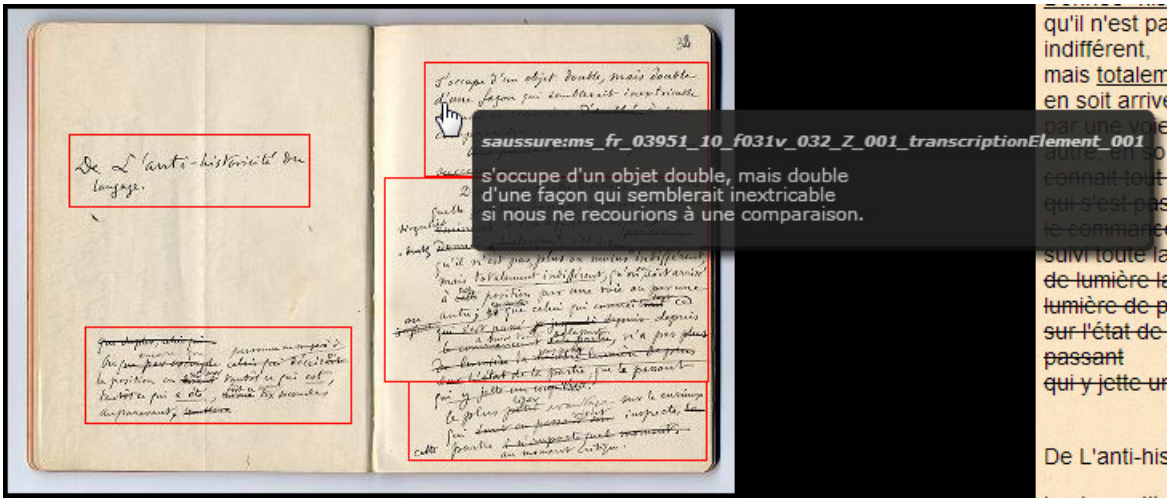


FIGURE 7.6 – *Tooltip cachant une zone*

Plusieurs solutions sont possibles pour contourner ce problème.
La première serait de complètement supprimer le *tooltip* et de lier un événement "simple clic" sur l'annotation. Le simple clic sélectionnerait l'annotation (en changeant la couleur de la bordure de la zone par exemple) et afficherait le contenu textuel dans une zone dédiée qui se situerait hors de l'image. La deuxième solution serait de lier un événement "double-clic", équivalant à celui sur la zone, au tooltip et éventuellement aux différents éléments affichés dans la zone d'affichage de la transcription (panneau de droite).

Notons encore, pour la bonne forme, que l'édition d'un élément de transcription, si celui-ci fait partie d'une transcription validée par un expert saussurien, ne devrait pas pouvoir être modifiée par n'importe quel utilisateur. Une telle modification ne devrait pas être effectuée très fréquemment.

7.2.2 Efficience

L'efficience est la capacité à atteindre le résultat avec un effort moindre ou dans un temps minimal. On a vu que l'efficacité du système est globalement bonne (taux de réussite global de 97%). Ici, on s'intéresse à l'effort requis pour l'exécution des tâches.

Les temps d'exécution de chaque tâche par chaque utilisateur se trouvent dans le tableau 7.7. Le graphique 7.8 montre les temps d'exécution moyens des tâches. Les tâches sont groupées par type (beige (1) : navigation avec les *thumbnails* ; bleu (5) : tâches de recherche ; orange (2) : tâches d'édition ; gris (5) : tâches de création).

Avant de continuer, rappelons qu'une tâche est une suite d'étapes que l'utilisateur doit lire, comprendre, puis exécuter (voir Annexe E.2). Certaines tâches sont plus longues à exécuter que d'autres, parfois les étapes sont plus ou moins compliquées. On a également pu constater que certains participants ont eu des problèmes, car ils ne lisaient pas entièrement la marche à suivre.

	U01	U02	U03	U04	U05	U06	U07	U08	U09	U10	U11	U12	U13	U14	Moyenne	Ecart-type
Scénario 1 - tâche 1	00:20	00:41	00:57	00:39	00:41	00:51	00:41	00:50	00:39	00:41	00:45	00:58	00:56	00:45	00:45	00:10
Scénario 1 - tâche 2	00:19	00:42	00:24	00:42	00:36	00:36	00:32	00:33	00:34	00:46	00:48	00:24	00:30	01:24	00:38	00:16
Scénario 2 - tâche 1.2	01:50	00:46	01:00	00:36	00:00	00:39	00:41	00:48	00:48	01:09	00:58	01:08	01:43	01:19	00:58	00:28
Scénario 2 - tâche 1.5	01:41	00:32	01:19	00:33	01:10	00:49	00:54	00:32	01:45	01:01	00:35	00:44	01:00	01:12	00:59	00:24
Scénario 2 - tâche 2	01:48	00:56	01:30	01:18	00:28	01:03	01:35	03:38	00:46	01:08	00:41	00:53	01:02	00:56	01:16	00:46
Scénario 2 - tâche 3	00:42	00:34	01:18	00:31	00:29	00:28	00:28	00:30	00:24	00:35	00:22	00:20	00:51	00:47	00:36	00:15
Scénario 2 - tâche 4	00:37	00:33	00:54	00:26	00:27	00:56	00:57	01:55	01:52	00:32	00:34	01:07	01:33	01:36	01:00	00:32
Scénario 2 - tâche 5	-	-	00:42	01:03	00:54	01:33	01:23	01:28	01:35	01:10	00:56	01:15	01:00	01:36	01:13	00:18
Scénario 3 - tâche 1	01:43	01:12	00:00	01:09	01:11	00:58	00:49	01:27	01:17	01:50	01:25	01:05	02:07	01:53	01:18	00:32
Scénario 3 - tâche 2	01:10	00:52	02:11	00:38	00:56	00:55	00:40	01:09	01:07	00:50	00:47	01:38	00:58	01:49	01:07	00:27
Scénario 3 - tâche 3	-	-	01:04	00:52	01:05	01:24	01:25	01:07	02:02	01:17	01:07	00:50	02:12	01:30	01:20	00:25
Scénario 4 - tâche 1	00:00	01:54	01:03	00:44	00:59	01:08	01:25	01:29	01:40	01:15	01:00	01:00	01:36	01:15	01:11	00:28
Scénario 4 - tâche 2	01:57	01:35	01:20	01:18	01:03	01:12	02:11	01:19	01:18	01:14	01:14	02:04	01:20	01:26	01:28	00:21

FIGURE 7.7 – *Tableau des temps*

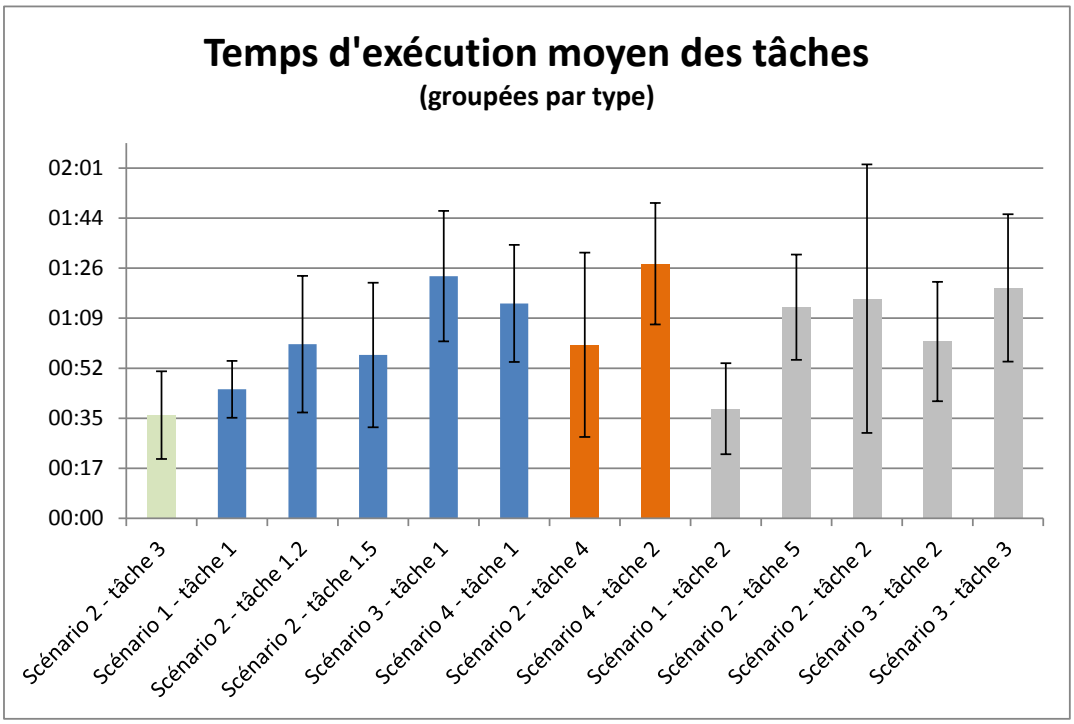


FIGURE 7.8 – *Temps d'exécution moyen des tâches*

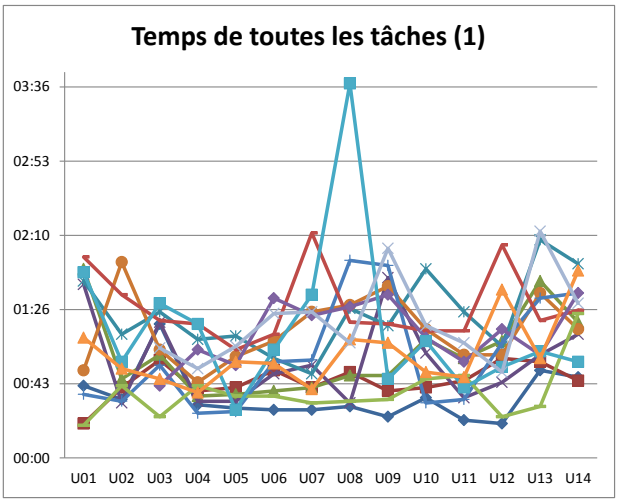
Du point de vue de la durée des tâches, l'efficience est globalement plutôt bonne. Le temps d'exécution moyen d'une tâche (toutes tâches confondues) est de 01m04s (inclus le temps de lecture, etc.).

Sur le graphique 7.8, on voit que la recherche par concept (S3-T1) a pris beaucoup de temps. Comme cette tâche demande de visualiser 3 manuscrits à la suite, il y a eu un peu de confusion (faut-il revenir en arrière ? comment revenir en arrière ? ...). Aussi, lorsque le bouton *Back* du navigateur est utilisé, un message s'affiche demandant à l'utilisateur de recharger la page (message standard lorsqu'une page utilise les données POST d'un formulaire). Bien qu'il soit expliqué dans les étapes de la tâche comment gérer ce cas, beaucoup d'utilisateurs ont pensé avoir fait une erreur et ont passé du temps à hésiter. Une solution consisterait à retrouver les

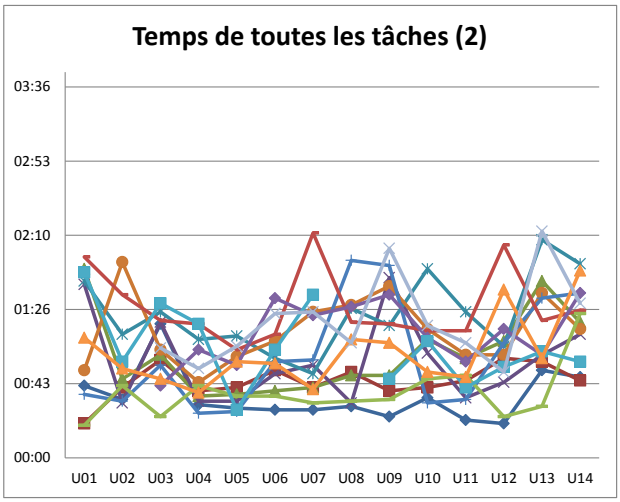
résultats de recherche lors de la visualisation du manuscrit (pour éviter de devoir revenir en arrière). On pourrait aussi contourner ce problème en incluant un bouton permettant de revenir à la page précédente. Il serait encore possible d'envoyer les données en GET plutôt qu'en POST, mais cette dernière solution n'est pas toujours réalisable.

La modification d'une transcription existante (S4-T2) est également une tâche longue. Comme expliqué plus haut, le taux de réussite de cette tâche n'est pas de 100%, et certains utilisateurs ont pris du temps à comprendre comment modifier la transcription. Du point de vue de l'efficience cela n'est pas dramatique, car cette opération ne devrait pas être très souvent utilisée et sera réservée à des utilisateurs disposant de droits particuliers.

La tâche "Lier un concept au manuscrit" (S2-T2) est la plus inquiétante, car l'écart-type est vraiment important. C'est sur cette tâche qu'il y a eu le seul cas d'échec à cause d'un dépassement du temps (3m38s). La moyenne pour cette tâche est de 01m16 et l'écart-type de 00m46. En ignorant le temps dépassé, on aurait une moyenne de 01m05 et un écart-type de 00m23. On voit sur la figure 7.9(a) les temps de chaque utilisateur pour toutes les tâches (une courbe par tâche). La figure 7.9(b) montre la même chose, mais en ignorant le temps trop long de 3m38s. Comme il s'agit d'un cas isolé, il peut aisément être ignoré.



(a) *Temps de toutes les tâches*



(b) *Temps de toutes les tâches en ignorant le temps dépassé*

Le dernier cas dont l'efficience nous interpelle est la création d'une transcription (S3-T3). Ce qui peut sembler étonnant, c'est que tous les utilisateurs ont fait deux tâches de création de transcription dans le même ordre (S3-T2, puis S3-T3). Or, S3-T2 semble avoir posé beaucoup moins de problèmes. Intuitivement, on pourrait s'attendre à ce qu'une notion d'apprentissage intervienne et diminue le temps entre deux tâches consécutives et de même nature. On constate alors le même phénomène avec la création de deux annotations (S1-T2 et S2-T5). Attention, ces deux cas ne sont pas exactement comparables, car contrairement à S3-T2 et S3-T3, tous les utilisateurs n'ont pas effectué ces tâches dans le même ordre. U01, U02, U03, U11, U14 ont exécuté le scénario 1 avant le scénario 2. Les neuf autres utilisateurs ont d'abord exécuté la tâche S2-T5. Là encore, le résultat semble contre-intuitif.

On peut en déduire que la notion d'apprentissage est négligeable, et que le fondement de ces écarts se situe au niveau de la tâche elle-même. L'explication est simple. Les tâches S3-T3 et S2-T5, qui ont demandé plus de temps à exécuter, sont un peu plus complexes que des tâches comparables (S3-T2 et S1-T2), car il est demandé de "zoomer deux fois sur l'image" avant de créer la zone. À ce moment, beaucoup d'utilisateurs ont zoomé, repéré la zone à transcrire, puis ont appuyé sur le bouton "Transcriptions" (voir figure 7.9) pour basculer dans le mode

"Transcriptions". Ceci a pour conséquence de réinitialiser la vue, et par conséquent on "perd" le niveau de zoom et la position. Certains utilisateurs l'ont même fait plusieurs fois. Notons aussi que, souvent, les utilisateurs cliquaient sur le bouton du mode souhaité avant de créer une zone, afin de s'assurer qu'ils étaient dans le bon mode. Cela ne posait pas de problèmes lorsque l'utilisateur n'avait pas zoomé.

FIGURE 7.9 – *Boutons de l'interface pour basculer d'un mode à l'autre*

Une amélioration possible consisterait à ignorer l'action sur le bouton s'il est déjà sélectionné. Il serait également possible, mais un peu plus compliqué, de garder l'information du niveau de zoom et de la position, puis de les récupérer lors du clic sur le bouton (cela permettrait par exemple de basculer du mode "Annotations Textuelles" vers le mode "Transcriptions" sans perdre sa position.

Le tableau 7.10 compare, pour chaque utilisateur, le temps total de son évaluation (somme des temps de chaque tâche), le score SUS (voir ci-dessous), ainsi que son taux de réussite.

	U01	U02	U03	U04	U05	U06	U07	U08	U09	U10	U11	U12	U13	U14
Fin calculée	12:16	09:43	12:24	09:58	09:30	12:04	13:13	16:15	15:23	12:53	10:50	13:06	15:57	16:41
Score SUS	92.5	82.5	80	97.5	95	95	92.5	95	85	100	97.5	92.5	87.5	75
Taux réussite	100%	95%	100%	100%	100%	100%	88%	85%	100%	100%	100%	88%	100%	100%

FIGURE 7.10 – *Tableau des ordres d'exécution des scénarios*

On ne peut malheureusement pas tirer beaucoup de conclusions de ce tableau. Le taux de réussite ne semble pas vraiment dépendre du temps : U07, U08 et U12 ont un taux de réussite plus bas que les autres, et des temps supérieurs à la moyenne. Par contre les utilisateurs U09, U13 et U14 ont des temps très élevés, mais un taux de réussite de 100%.

Il en va de même pour la corrélation entre le taux de réussite et le score SUS. Les utilisateurs U02, U03, U09, U13 et U14 ont un score SUS plus bas que la moyenne. Cependant, U03, U09, U13 et U14 ont tous un taux de réussite de 100%.

Il n'y a pas vraiment de corrélation non plus entre le temps et la satisfaction. Les utilisateurs ayant un temps supérieur à la moyenne ont, dans les mêmes proportions, des scores SUS en dessous de la moyenne (U09, U13, U14), et au-dessus (U07, U08, U12).

L'âge des participants, leurs compétences en linguistique ou leur rapport aux manuscrits ne semblent pas non plus être une source d'analyse intéressante pour tirer des conclusions.

Il y a peut-être une influence des compétences informatiques de l'utilisateur sur la satisfaction. Les utilisateurs U02, U08, U09 et U14 ont répondu de façon plutôt négative à l'affirmation "Je suis à l'aise avec un ordinateur". On remarque qu'il n'y a pas d'influence sur le taux de réussite, mais que U02, U09 et U14 ont un score SUS en dessous de la moyenne.

7.2.3 Satisfaction (*System Usability Scale* (SUS))

La satisfaction est définie comme le confort et l'évaluation subjective de l'interaction par l'utilisateur. On la mesure grâce aux résultats du formulaire SUS.

Le *System Usability Scale* (SUS) [5] est un simple formulaire [6] ("*quick and dirty*" comme le dit John Brooke, son auteur) d'une dizaine de questions, qui permet d'évaluer la satisfaction de l'utilisateur. Chaque question peut recevoir une note sur une échelle de 1 (*Pas du tout d'accord*) à 5 (*Tout à fait d'accord*) ; si l'utilisateur pense que la réponse ne s'applique pas ou qu'il ne peut pas y répondre, il doit noter 3. Les notes sont converties selon une formule pour obtenir un score (et non pas un pourcentage) entre 0 et 100.

Calcul du score SUS :
L'échelle utilisée est une échelle sur 5 (certaines versions existent avec une échelle sur 7).
Pour les items 1, 3, 5, 7 et 9, le score est le résultat moins 1. (tout à fait d'accord : $5-1=4$)
Pour les items 2, 4, 6, 8 et 10, le score est 5 moins le résultat. (tout à fait d'accord : $5-5=0$)
Faire le total des scores et multiplier par 2,5 pour obtenir le score SUS qui varie de 0 à 100.

Cette technique d'évaluation de l'utilisabilité est simple, rapide et a fait ses preuves depuis des années (elle a été développée au milieu des années 80, mais publiée en 1996).

Malgré sa popularité, l'article proposant le formulaire SUS n'explique pas réellement comment interpréter un score SUS. Une étude a été réalisée sur l'interprétation des scores SUS [7]. Cette étude a analysé les résultats SUS d'environ 500 produits (à savoir un total d'environ 5000 utilisateurs) et conclut que les scores peuvent être interprétés comme ceci :

- Un score de 68 est défini comme la moyenne.
- En dessous de 68, il faut commencer à remettre en question le *design* de l'application.
- Un score supérieur à 68 est donc positif.
- Un score supérieur à 80.3 est très positif (parmi les 10% des meilleurs scores).

Les questions du formulaire sont :

- **SUS01** Je pense que je vais utiliser ce service fréquemment.
- **SUS02** Je trouve ce service inutilement complexe.
- **SUS03** Je pense que ce service est facile à utiliser.
- **SUS04** Je pense que j'aurai besoin de l'aide d'un technicien pour être capable d'utiliser ce service.
- **SUS05** J'ai trouvé que les différentes fonctions de ce service ont été bien intégrées.
- **SUS06** Je pense qu'il y a trop d'incohérences dans ce service.
- **SUS07** J'imagine que la plupart des gens seraient capables d'apprendre à utiliser ce service très rapidement.
- **SUS08** J'ai trouvé ce service très lourd à utiliser.
- **SUS09** Je me sentais très en confiance en utilisant ce service.
- **SUS10** J'ai besoin d'apprendre beaucoup de choses avant de pouvoir utiliser ce service.

Le questionnaire alterne le ton des questions (les questions impaires sont positives, les paires sont négatives), les réponses doivent donc subir un traitement pour être comparées (comme dans la formule de calcul du score (voir plus haut)). Pour avoir un "bon" résultat, une question positive (impaire) doit avoir la note maximum (5), et une question négative (paire) doit au contraire avoir la note minimale (1).

5. `http://hell.meiert.org/core/pdf/sus.pdf`
6. `http://blocnotes.iergo.fr/concevoir/les-outils/sus-pour-system-usability-scale/`
7. Le contenu de l'étude est payant, mais les résultats sont expliqués sur le site web : `http://www.measuringusability.com/sus.php`

Le tableau 7.11 présente les réponses des utilisateurs au questionnaire SUS. Il en ressort assez clairement que les réponses étaient positives dans l'ensemble (valeurs en vert). Dans toutes les réponses confondues, il n'y a eu qu'une seule réponse négative à la question SUS09 de l'utilisateur U09. Cet utilisateur a pourtant eu un taux de réussite de 100%. Il a par contre donné une note de 2 à la question "Je suis à l'aise avec un ordinateur". Sans remettre en question cette valeur, on peut se demander si le sentiment de "confiance" de SUS09, n'a pas été amalgamé à un *a priori* de l'utilisateur quant à sa capacité à utiliser un ordinateur.

	U01	U02	U03	U04	U05	U06	U07	U08	U09	U10	U11	U12	U13	U14	Moyenne	Moyenne normalisée
SUS01	3	5	3	5	3	3	4	5	3	5	4	5	3	4	3.93	3.93
SUS02	1	2	1	1	1	1	1	1	1	1	1	1	1	2	1.14	4.86
SUS03	5	4	4	5	5	5	4	5	3.5	5	5	4	4	4	4.46	4.46
SUS04	1	1	3	1	1	1	1	1	1	1	1	2	1	1	1.21	4.79
SUS05	5	5	4	5	5	5	5	5	5	5	5	5	5	4	4.86	4.86
SUS06	1	1	2	1	1	1	1	1	1	1	1	2	1	2	1.21	4.79
SUS07	4	3	5	4	5	5	5	5	5	5	5	5	4	3	4.50	4.50
SUS08	1	3	1	1	1	1	1	1	1	1	1	1	1	2	1.21	4.79
SUS09	5	4	4	5	5	5	4	4	2.5	5	5	5	4	4	4.39	4.39
SUS10	1	1	1	1	1	1	1	2	1	1	1	1	1	2	1.14	4.86
Score SUS	92.5	82.5	80	97.5	95	95	92.5	95	85	100	97.5	92.5	87.5	75	90.54	

Réponse positive Neutre ou ne s'applique pas Réponse négative

FIGURE 7.11 – *Tableau des résultats SUS*

Afin de pouvoir comparer dans un graphe les valeurs alternées des questions, on "normalise" les valeurs sur 5. Les notes des questions impaires sont préservées, alors que les notes des questions paires sont soustraites à 6 pour obtenir une note sur 5 (par exemple la note 2 à la question 2 : $6 - 2 = 4$).

Le graphique 7.12 compare les résultats moyens, normalisés sur 5, de chaque question du formulaire. Comme les utilisateurs ne travaillent pas tous régulièrement avec des manuscrits, six d'entre eux (soit 43% des utilisateurs) ont mis la note 3 (= "ne s'applique pas") à la première question. Le résultat de cette question est donc biaisé, car le système testé vise un public spécifique (il s'agit presque d'une application métier) et on ne s'attend pas à ce que tous les testeurs souhaitent l'utiliser réellement. Afin d'avoir une meilleure idée du résultat pour les personnes concernées, le graphique 7.13 compare les résultats moyens de chaque question du formulaire normalisés sur 5, en ignorant les réponses neutres de la première question (et de la première question uniquement). Finalement, le graphique 7.14 montre le score SUS de chaque utilisateur.

La satisfaction est plutôt bonne avec des scores moyens élevés par question. En ignorant les résultats neutres de la question SUS01 (voir graphique 7.13), le score normalisé le plus bas est 4.39, ce qui est tout de même très bon.

Les scores pour chaque utilisateur (graphique 7.14) sont plus variés. Les résultats vont de 75 à 100, avec une majorité (9 utilisateurs) au-dessus de 90.

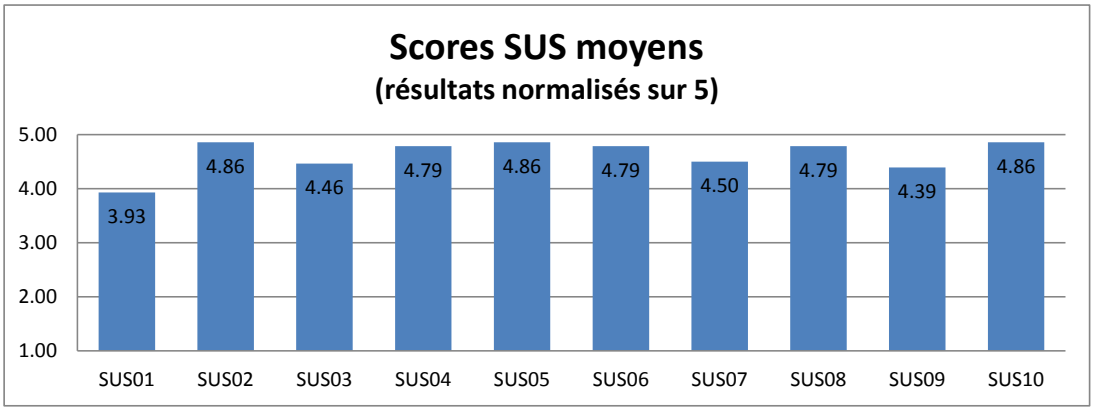


FIGURE 7.12 – *Scores SUS normalisés moyens*

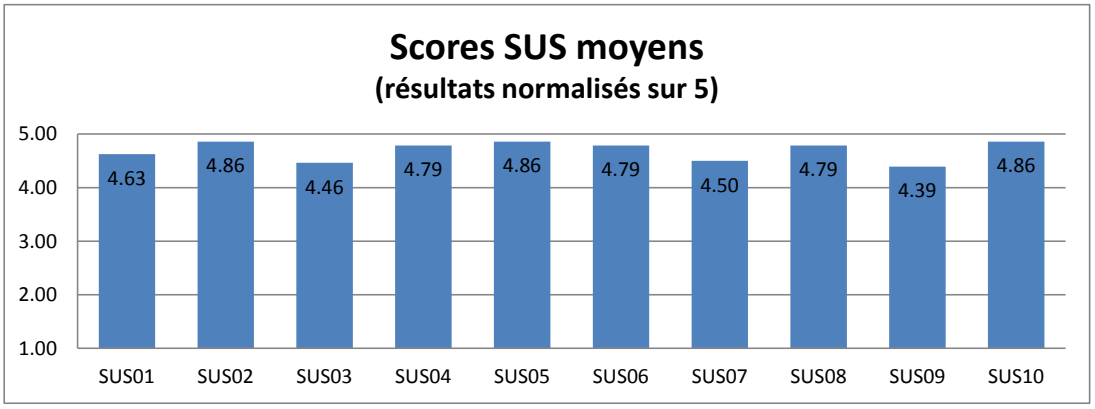


FIGURE 7.13 – *Score SUS normalisé moyens (en ignorant les "ne s'applique pas" de SUS01)*

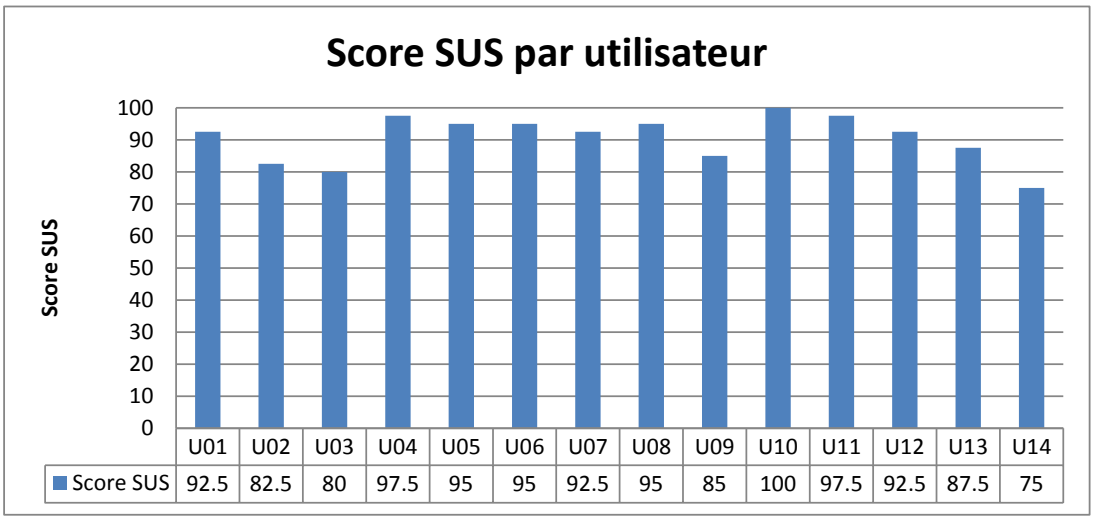


FIGURE 7.14 – *Score SUS par utilisateur*

7.2.4 Commentaires, propositions d'amélioration, remarques des participants

Cette section résume, pêle-mêle, les commentaires ou remarques des utilisateurs :

- **U01**
  - Propose d'afficher plus clairement la cote du manuscrit. A indiqué qu'il cherchait cette information en haut de la page.
  - A trouvé perturbant que le simple clic ne soit lié à rien sur les zones. Proposerait un menu contextuel proposant l'effacement par exemple.
  - A trouvé perturbant que CTRL+Scroll ne zoome pas sur le manuscrit.
- **U02**
  - Positif sur la fluidité.
  - N'a pas trouvé que l'icône "rectangle" de la *toolbar* reflète l'action de créer une zone.
  - N'est pas convaincu par la zone d'affichage de la transcription (panneau de droite).
  - Trouve que l'édition d'une transcription devrait se faire côte à côte et prendre beaucoup plus de place.
  - Souhaiterait accéder au catalogue de ce qui est en ligne.
  - Pense qu'un curieux (utilisateur lambda) ne trouverait pas facilement l'information.
- **U03**
  - Aucune remarque.
- **U04**
  - Aimerait voir une transcription ligne par ligne, avec une numérotation des lignes.
  - Aimerait pouvoir exporter la transcription vers Word ou autre.
  - Aimerait pouvoir imprimer le manuscrit.
  - Propose que l'interface ne perde pas le niveau de zoom et la position lors du basculement de mode.
  - Aimerait pouvoir créer des annotations et des transcriptions et choisir de les partager ou non. Par exemple, l'effort de réaliser une transcription n'est pas forcément quelque chose que l'on souhaite partager avant que son étude ne soit terminée.
  - N'a pas aimé devoir rafraîchir la page (F5) lors du retour sur les résultats de recherche.
  - Aimerait voir un index/inventaire avec le contenu et si possible avec la date des manuscrits.
  - Propose d'ajouter quelque chose au carrousel (navigation par *thumbnail*) pour indiquer où l'on se situe, comme un ascenseur.
- **U05**
  - Aucune remarque.
- **U06**
  - A trouvé très convivial.
  - A trouvé utile d'avoir les *titles* (bulles d'information lorsqu'on met la souris dessus) sur les boutons de la *toolbar*.
- **U07**
  - Aucune remarque.
- **U08**
  - "Une fois qu'on a fait et compris, c'est très facile et il y a plus de potentialités qu'on aurait imaginé".
- **U09**
  - Aucune remarque.
- **U10**
  - A trouvé peu clair de devoir rafraîchir la page (F5) lorsqu'il faut revenir sur un résultat de recherche. Propose de retrouver les résultats de recherche sous une forme ou une autre dans la page qui affiche le manuscrit.

- **U11**
  - A trouvé très intuitif.
  - A trouvé que la disposition est claire, que les différentes parties sont bien séparées, et que ce qui doit être ensemble est bien groupé.
  - A trouvé que chaque option est facilement accessible (pas besoin d'aller dans des menus, l'interface ne change pas radicalement selon nos actions) et que l'on trouve rapidement ce que l'on cherche.
  - A approuvé la navigation selon l'ordre des archives, notamment la notion de subdivision telle qu'elle a été définie dans ce travail (voir la section 4.1).
- **U12**
  - A trouvé qu'un catalogue est indispensable pour savoir ce que contient le site. Ce que le site contient est "la liste des cotes".
- **U13**
  - Aucune remarque.
- **U14**
  - "Travail très précis".
  - Propose que l'affichage de la cote soit plutôt vers le haut de la page ou de l'image.

7.3 Bilan

Ce chapitre a présenté une évaluation globale du système, autant en termes de satisfaction que de recherche d'erreurs.
Les outils utilisés (formulaires, scénarios, formules de calcul, etc.) sont ceux qui semblaient le mieux adaptés à ce travail. Ce chapitre n'a pas la prétention d'explorer toutes les solutions en matière d'utilisabilité. Par exemple, il existe des moyens de déterminer numériquement à l'avance le temps d'exécution d'une tâche [8] par un utilisateur expérimenté, ce qui est utile pour fixer le temps limite d'une tâche. Il aurait aussi été possible d'évaluer le phénomène d'apprentissage en répétant des tâches similaires dans les scénarios, pour estimer la vitesse à laquelle un utilisateur apprend à utiliser l'interface. Ce type d'analyse sort largement du cahier des charges de ce travail ou n'est simplement pas possible avec les mesures en notre possession.

Les tests utilisateurs ont permis de mettre en avant les forces et faiblesses de l'interface. Cependant, l'analyse n'a pas permis de faire ressortir une information très intéressante en matière de profils d'utilisateurs ou de corrélation entre les résultats.

Globalement, les résultats sont bons, voire très bons, et en tout cas bien meilleurs qu'escomptés pour une première évaluation du système. Le taux d'efficacité global de 97% est excellent. La satisfaction est aussi très bonne avec un score SUS moyen de 90.5. D'après l'étude citée dans la section 7.2.3, un score SUS meilleur que 80.3 fait partie des "*10% meilleurs systèmes*". Le système fait donc partie encore de la tranche supérieure de ces 10%. Un autre article, proposant la conversion de scores SUS en adjectifs [9], assigne l'"adjectif" *Best imaginable* aux scores supérieurs à 90.

Malgré ces excellents résultats, il y a cependant des choses à améliorer en terme d'efficience (mais le contraire aurait été suspect). En effet, les résultats du questionnaire SUS, les commentaires des participants et l'analyse faite dans ce chapitre permettent de mieux cibler les corrections qu'il reste à effectuer.

8. `http://www.measuringusability.com/predicted-times.php`
9. `http://www.upassoc.org/upa_publications/jus/2009may/JUS_Bangor_May2009.pdf`

7.3.1 Modifications/Améliorations proposées

Sur la base des commentaires, et des constatations relevées dans l'analyse des résultats, voici une liste des améliorations proposées pour pallier les quelques problèmes d'interface rencontrés.

- **Problème du *tooltip* qui cache la zone (confusion dans l'édition d'une transcription).**
  Solution 1 Complètement supprimer le *tooltip* et lier un événement "simple clic" sur l'annotation. Le simple clic sélectionnerait l'annotation (en changeant la couleur de la bordure de la zone, par exemple) et afficherait le contenu textuel dans une zone dédiée qui se situerait hors de l'image.
  Solution 2 Lier un événement "double-clic", équivalant à celui sur la zone, au tooltip et éventuellement aux différents éléments affichés dans la zone d'affichage de la transcription (panneau de droite).
- **Problème pour revenir à la page de recherche.**
  Solution 1 Faire en sorte de retrouver les résultats de recherche lors de la visualisation du manuscrit (pour éviter de devoir revenir en arrière).
  Solution 2 Sur la page du manuscrit, inclure un bouton permettant de revenir à la page précédente et qui contournerait ce problème.
  Solution 3 Envoyer les données en GET plutôt qu'en POST. Attention : cette solution n'est pas toujours possible.
- **Problème de réinitialisation lors du clic sur les boutons pour basculer de mode.**
  Solution 1 Ignorer l'action sur le bouton s'il est déjà sélectionné.
  Solution 2 Garder l'information du niveau de zoom et de la position et les récupérer lors du clic sur le bouton (cela permettrait, par exemple, de basculer du mode "Annotations Textuelles" vers le mode "Transcriptions" sans perdre sa position).
  Solution 3 Modification du *design* pour remplacer l'action du bouton "dessiner un rectangle" par un menu proposant de créer les différents types de zone ("Annotation textuelle", "Annotation conceptuelle", "Transcription"). À ce moment, il peut être utile de remplacer le comportement des boutons au bas de l'image : toutes les annotations, transcriptions et annotations conceptuelles seront chargées en même temps et les boutons ne serviront qu'à afficher/masquer les annotations d'un certain type. Ceci permettrait d'éviter une confusion supplémentaire lorsque l'utilisateur se positionne, puis clique sur l'un des boutons en dessous de l'image (pour confirmer qu'il veut bien une annotation textuelle, par ex.), car ceci réinitialise la vue (perte du zoom et de la position). Mais en appliquant les modifications proposées, il n'y aurait plus de réinitialisation de la vue (uniquement le *toggle* d'affichage de certaines zones).

Chapitre 8

Conclusion

8.1 Travaux futurs

Ce travail de Master n'avait pas la prétention de couvrir l'ensemble des sujets abordés lors de l'analyse des besoins des utilisateurs. Une grande partie des tâches ont cependant été implémentées, mais il reste encore du développement à faire pour ajouter des fonctionnalités ou améliorer des cas détectés comme problématiques.

Cette section décrit les propositions de correction ou d'amélioration. Celles-ci découlent de l'analyse des besoins, de suggestions spontanées de l'auteur, ainsi que des évaluations réalisées avec les utilisateurs (soit par observation, soit lors de la discussion suivant l'évaluation, ou encore dans les remarques figurant dans les formulaires).

Du point de vue de l'interface :

- Lors de l'évaluation, plusieurs utilisateurs ont émis le souhait de pouvoir transcrire le manuscrit avec l'éditeur et l'image côte à côte, plutôt que l'éditeur en dessous de l'image. Il a également été question de pouvoir faire une transcription ligne par ligne. Ces propositions sont citées ici, car elles sont communes à plusieurs utilisateurs. Cependant, le système proposé a pour volonté d'offrir une interface simple et facile à utiliser. La transcription ligne par ligne est contraignante si l'utilisateur doit sélectionner manuellement les lignes. On peut aussi remarquer que cette façon de faire relève aussi d'une expérience préalable dans le domaine, car les outils existants utilisent souvent ce type de saisie (par exemple, T-Pen - voir la section 3.4.4.2).
- Afin de faciliter l'accès au contenu, un catalogue des manuscrits en ligne est indispensable. Celui-ci devrait être lié à celui de la BGE.
- Indiquer la cote du manuscrit en haut de la page ou au-dessus de l'image.
- Ne pas réinitialiser la vue lors du clic sur les boutons pour basculer d'un mode à un autre (voir la section 7.3.1).
- Éviter de devoir recharger la page lorsque l'utilisateur revient sur la page des résultats de recherche (voir la section 7.3.1).
- Par défaut, cacher la navigation (mini-map) sur l'image, car elle n'apporte pas grand chose. Il restera évidemment possible de l'afficher grâce à la barre d'outils.
- Un ascenseur dans le slider de *thumbnails* permettrait de savoir où l'on se situe.
- Créer des annotations lorsque l'image est pivotée. En effet, il s'agit d'un réel besoin, car sur certains manuscrits, le texte est réellement écrit dans tous les sens.
- L'ajout d'un système de filtre pour les annotations permettrait d'afficher uniquement les annotations d'un certain utilisateur, celles postérieures à une certaine date, etc.
- Une possibilité évoquée par les utilisateurs du projet était d'avoir une pseudo-synchronisation

entre l'image et sa transcription. La pseudo-synchronisation permettrait de situer la portion de l'image en cours de visualisation et d'aligner grossièrement la portion de transcription correspondante dans le panneau de droite.
- Il serait souhaitable de pouvoir visualiser plusieurs manuscrits simultanément, afin de les comparer (s'ils traitent d'un même sujet, par exemple).

Du point de vue du développement :
- Les scripts PHP pourraient gérer l'historique des modifications, et éventuellement des effacements. Ce type de système (comme dans MediaWiki par exemple) permet de limiter les abus lorsque les données sont publiques.
- Le serveur devrait gérer plusieurs niveaux d'authentification. Pour l'instant, on vérifie seulement si l'utilisateur est authentifié ou non. Il faudrait ajouter des niveaux d'autorisation.
- Il serait judicieux d'ajouter une fonction pour créer des transcriptions en spécifiant une séquence d'éléments de transcription. Celle-ci pourra être liée au formulaire en annexe (voir Annexe B.6.2) et ne pourra pas être exécutée par tous les utilisateurs (besoins d'un niveau d'autorisation élevé).
- De porter le projet pour une utilisation sur tablette tactile (*tablet*) et/ou *smartphone* (fonctionnalités possiblement réduites ou adaptées au support) pourrait être apprécié par certains.
- La possibilité de créer des annotations sur le texte transcrit, plutôt que sur l'image du manuscrit, pourrait également être utile.
- Un système de modération peut être envisagé afin de limiter le nombre ou le contenu des annotations. Seules les annotations validées par un modérateur seraient effectivement rendues publiques.

Du point de vue de l'aspect web social et *crowdsourcing* :
- *Crowdsourcing*/automatisation de la tâche de transcription : comme la tâche de transcription est un travail colossal, il est souhaitable qu'elle puisse être automatisée (du moins partiellement). L'utilisation d'un système d'OCR est exclue (cette solution avait déjà été explorée par les utilisateurs du projet). En effet, les manuscrits de Saussure comportent souvent des ratures, des insertions, et sont même parfois rédigés dans plusieurs sens (la feuille est pivotée lors de la rédaction). Un système d'OCR personnalisé pourrait être réalisé, même si les chances de succès sont maigres. La meilleure solution consiste à utiliser du *crowdsourcing*, soit en invitant des volontaires à participer (comme le **Bentham Project** [1]), soit en les y "obligeant" (comme avec reCaptcha [2] qui numérise des livres en soumettant des *captchas* aux utilisateurs). Il serait aussi possible de contacter Google/reCaptcha afin de leur proposer de soumettre les archives de Saussure dans les *captchas* de reCaptcha.
- Approche "web social" pour la gestion du contenu du serveur. La couche d'authentification mise en place permet de facilement faire évoluer la plateforme vers le web social. En effet, lors de l'évaluation de l'interface, plusieurs utilisateurs ont précisé qu'ils préféraient pouvoir contrôler la visibilité de leurs annotations ou de leurs transcriptions. Du moment que les utilisateurs sont identifiables, et que le contenu des annotations/transcriptions aussi (c'est déjà le cas), il est tout à fait possible d'étoffer la granularité au niveau des requêtes. On pourrait, par exemple, proposer des groupes de travail et, lors de la création d'une annotation, choisir :
  - visible pour moi
  - visible pour le groupe "Cercle Ferdinand de Saussure"
  - visible pour tout le monde

1. `http://www.ucl.ac.uk/Bentham-Project`
2. `http://www.google.com/recaptcha`

8.2 Conclusion

Le but de ce travail consistait à mettre en ligne des manuscrits numérisés de Ferdinand de Saussure sous la forme de photographies numériques en haute résolution. Le système développé devait permettre aux linguistes de travailler sur le document de manière souple, sans les contraintes d'une visite à la bibliothèque ou celles relatives à la manipulation de matériel d'archives (souvent difficile d'accès, voire impossible dans le cas des manuscrits originaux de Ferdinand de Saussure). À cet effet, il fallait notamment offrir des outils interactifs pour la consultation, mais également pour l'annotation et la transcription des manuscrits. Les objectifs, encore mal définis au début du projet, ont été affinés en accord avec les utilisateurs grâce à un important travail d'analyse des besoins effectué lors de nombreuses réunions avec ceux-ci.

Un état de l'art portant sur les projets de recherche similaires, ainsi que sur un large éventail de problématiques techniques a permis d'avoir une vue d'ensemble pour proposer les solutions qui semblaient les plus judicieuses.

Le modèle de données ainsi qu'une interface utilisateur (interface graphique) ont découlé de l'analyse des besoins des utilisateurs, et ont permis la réalisation d'un prototype fonctionnel. Celui-ci contient toutes les fonctionnalités souhaitées par les utilisateurs, et consiste en un logiciel complet gérant tous les aspects techniques du projet, depuis la mise en ligne des images, jusqu'aux interactions de l'utilisateur grâce à une application web.

Un modèle de données clair, structuré, mais prévoyant les évolutions futures, a ainsi été conçu. Il intègre les données (et métadonnées) représentant les documents physiques de la bibliothèque (typiquement leur emplacement et le système de classification de la BGE), ainsi que les données relatives à l'images (la cote, les zones, les annotations, etc.). Toutes les données sont stockées sous forme de triplets RDF dans un triplestore OpenRDF Sesame. Celui-ci est accessible en ligne grâce à un *endpoint* SPARQL permettant la publication des données sur le web sémantique.

Un moteur de recherche personnalisé a été réalisé sur la base du modèle de données afin de permettre de trouver l'information liée aux manuscrits dans le triplestore. Il permet la recherche par mots-clefs (*full text search*) sur la cote, ainsi que sur le contenu des transcriptions, des annotations textuelles et des annotations conceptuelles.

Un script d'automatisation de la mise en ligne des images devra permettre d'alimenter très facilement le contenu du site avec de nouvelles images. Les images sont converties dans le format adéquat (JPEG2000), l'image originale est archivée et toutes les métadonnées sont insérées automatiquement dans le triplestore en respectant le modèle de données.

Plusieurs extensions ont été développées pour le projet IIPMooViewer, utilisé pour l'affichage de l'image du manuscrit. Les plus significatives sont l'ajout d'une barre d'outils et l'amélioration du système d'édition des annotations. Le code source de ces deux extensions est d'ailleurs en cours de publication afin d'être intégré dans le projet IIPMooViewer original. Cette contribution à la communauté *open source* assurera le maintien de la compatibilité du code et de ces fonctionnalités avec IIPMooViewer. Cela permettra à l'application développée dans ce projet de profiter des mises à jour d'IIPMooViewer, ce qui lui assurera par la même occasion une certaine pérennité.

Une application Javascript personnalisée a été développée pour les besoins du projet. Outre la gestion des communications avec le service web (frontal), celle-ci intègre notamment une interface utilisateur aboutie qui a été évaluée par un nombre important de volontaires. Les résultats de ces évaluations ont été plus que positifs et ont permis de mettre en avant deux choses importantes en terme d'utilisabilité : non seulement le système fonctionne, mais en plus

il est utilisable par des utilisateurs qui ne sont ni des experts, ni familiers avec le projet. En effet, les résultats prouvent que l'interface est facilement utilisable (efficience), mais également que le système permet d'atteindre les résultats prévus (efficacité). Les utilisateurs ont été globalement très positifs sur la qualité du système (satisfaction).

Les recherches et les réalisations (le modèle de données et le prototype) effectuées dans ce projet offrent une base solide pour l'ajout de nouvelles fonctionnalités. La structure et l'organisation du code source permettent de reprendre le projet ou d'en modifier certaines parties aisément. Par ex. : il est possible de modifier l'affichage sans devoir toucher au modèle de données ou au protocole d'échange avec le service frontal. Ceci devrait aussi permettre à ce projet de continuer à vivre et évoluer, notamment, par l'intermédiaire des contributions au projet IIPMooviewer.

En conclusion, des solutions ont été trouvées et développées pour toutes les problématiques de ce travail. Le résultat de ce projet répond donc aux besoins des utilisateurs finaux en leur offrant un outil scientifique complet et original, autant au niveau du système en tant que tel, que du modèle de données, de l'interface utilisateur, ou des contributions au projet IIPMooViewer. L'utilisation de RDF permet la publication des données sur le web sémantique, offrant ainsi des possibilités de lien avec d'autres ressources ou ontologies.

Dans une certaine mesure, on espère que le champ d'utilisation des contributions s'étendra à d'autres domaines qui utilisent aussi des manuscrits numérisés (histoire), mais pas seulement. En effet, ces développements pourraient s'appliquer à n'importe quel type d'images : photographies de peintures (histoire de l'art), d'objets ou de pièces (archéologie), etc. Ils pourraient également s'appliquer en dehors des sciences humaines (astronomie, médecine). Les chercheurs de ces différents domaines pourraient aisément profiter du développement d'outils adaptés à leurs besoins spécifiques, en suivant une démarche comparable à celle des linguistes saussuriens.

Annexe A

Cahier des charges

A.1 Version 1

Travail de MA "Système de visualisation et d'annotation des documents numérisés du Fonds Saussure"

Ce travail de MA s'inscrit dans un projet global dont le but est de mettre en place une infrastructure hypermédia ouverte pour l'édition scientifique des travaux du linguiste Ferdinand de Saussure. Le but de ce travail de MA est de mettre en place un prototype qui se présentera sous la forme d'un site Web permettant :

- de mettre en ligne les reproductions photographiques en haute résolution des manuscrits de Saussure et d'autres documents relatifs (notes de ses étudiants, documents historiques, etc.)
- d'annoter une reproduction globalement avec du texte et / ou des attributs
- de délimiter graphiquement (par exemple à l'aide de rectangles) des zones des reproductions et de les annoter avec du texte et / ou des attributs
- de faire des transcriptions des reproductions

Le projet se focalisera sur l'évaluation des choix techniques, sur la conception des interfaces utilisateur et sur la réalisation d'un premier prototype implémentant les fonctionnalités énumérées ci-dessus. Un sous ensemble des documents de Saussure servira de corpus test pour permettre à l'équipe genevoise de mettre au point ce prototype.

Liste des choses à faire :

- état de l'art dans les logiciels d'annotation, également des systèmes voisins comme les GIS
- études des solutions techniques
- analyse des tâches des utilisateurs
- prototypage des interfaces utilisateur (prototype papier, powerpoint, ...)
- conception de la structure des données : reproduction, annotation, transcription, index, attributs
- implémentation du prototype
- amélioration du prototype en fonction des tests des utilisateurs

Suite du projet (au-delà du mémoire de MA) :

La suite du projet aura pour objectif l'enrichissement de ce système avec différents types d'index : des mots clé, des expressions à mots multiples, des index sémantiques (faisant référence à une ontologie de la linguistique saussurienne). Ces index seront en grande partie constitués automatiquement sur la base de la masse documentaire en utilisant des techniques de traitement automatique du langage développé au LATL et sur des modèles d'analyse pragmatique du

discours. Cet environnement de recherche innovant permettra des recherches philologiques approfondies sur les travaux de Saussure et sera proposé à la communauté de recherche saussurienne.

Remarque : comme le prototype réalisé dans le cadre de ce MA sera le point de départ d'un système plus ambitieux, il est important d'avoir une bonne compréhension du projet global afin de réaliser un prototype évolutif. Une analyse aboutie du projet est donc nécessaire avant de se lancer dans la conception du prototype. Ce travail d'analyse se fera de conserve avec les auteurs du projet.

A.2 Version 2

Cahier des charges pour le projet

Transcription et annotation des manuscrits de Ferdinand de Saussure

G. Faquet, L. Nerima
22 Novembre 2012

1. Contexte

Ce travail s'inscrit dans le cadre d'un projet qui vise à développer des modèles et des systèmes de support au travail scientifique sur les manuscrits de Ferdinand de Saussure (environ 50'000 feuillets). Une petite partie de ces manuscrits a déjà été numérisée sous forme de photos à haute définition. Les experts saussuriens aimeraient pouvoir travailler en ligne sur ces manuscrits et donner accès à ces manuscrits transcrits et annotés au public.

Les principales fonctions visées pour ces systèmes sont : la visualisation, la transcription et l'annotation en ligne des manuscrits; une chaine de traitement (workflow) pour l'alimentation du serveur d'images à partir des photos originales (un documentaliste ou scientifique en possession de photos de manuscrits doit pouvoir les envoyer de manière simple vers le serveur); la publication sur le web sémantique de la base d'images, transcriptions et annotations ("endpoint" SPARQL).

Ce projet comprend un volet théorique dans le domaine de la représentation des connaissances concernant des documents manuscrits. Il comprend ensuite un volet de développement, qui servira également à valider une partie du travail théorique.

2. Problématique en représentation des connaissances

Il s'agit de créer un modèle de représentation d'un corpus de manuscrits qui permettant d'effectuer des tâches de recherche scientifique (études Saussuriennes en l'occurrence). Le modèle devra prendre en compte différentes dimensions :

- l'image du manuscrit

- le contenu textuel brut (ordre "typographique")
- le contenu textuel réorganisé sous forme d'unités logiques cohérentes (transcriptions)
- la sémantique du texte (concepts référencés, entités nommées (personnes, lieux, ...), relations entre concepts, ...)
- la représentation des résultats d'études scientifiques, sous forme d'annotations structurées ou non, portant sur des éléments de manuscrits et pouvant toucher des aspects graphiques (encre utilisée, ratures, couleurs, ...) aussi bien que théoriques (linguistiques ou historiques)

Le modèle devra être suffisamment expressif pour servir à la spécification de traitements à usage scientifiques, en particulier la recherche d'information

- "full text" dans le corpus transcrit
- conceptuelle (concepts linguistiques ou autres)
- dans les annotations, soit textuelle, soit conceptuelle

L'un des défis est de créer un modèle suffisamment simple pour que le processus d'indexation (saisie de transcriptions et annotations) soit faisable en un temps raisonnable et avec un apprentissage court.

La définition de ce modèle doit s'appuyer sur un état de l'art portant sur

- les modèles de représentation de manuscrits
- les modèles d'indexation sémantique de documents (en particulier pour le web sémantique)

Validation

Pour valider l'expressivité du modèle on établira un ensemble de cas complexes de transcription et d'annotation et on montrera comment les exprimer dans le modèle. On montrer également comment spécifier des opérations de recherche d'information typiques.

Pour valider l'efficacité et la simplicité du modèle on réalisera un système informatique comprenant des fonctions de stockage et visualisation d'images de manuscrits, de saisie et modification de transcription et d'annotations et de recherche dans les transcriptions et annotations. On effectuera des tests avec des utilisateurs pour démontrer

- qu'ils arrivent à réaliser les tâches de saisie et modification
- qu'ils peuvent effectuer des recherches dans la base de connaissance

3. Problématique de développement

Il s'agit de développer un système de stockage, annotation et de recherche dans une base de manuscrits.

Fonctions du système

Du point de vue des utilisateurs (spécialistes Saussuriens et public) le système devra permettre de réaliser les tâches suivantes

- accès à un feuillet de manuscrit
    - sélectionner le feuillet
        - (différentes techniques à définir)
        - par numéro
        - par navigation dans des listes
        - par mots clés à trouver dans les transcriptions et annotations
    - afficher la photo, avec possibilité de zoom
    - afficher la transcription, si elle existe
    - afficher les annotations

Contrainte: le système doit être rapide, il n'est pas concevable d'envoyer à chaque fois des images de 50Mb. il faudra donc mettre en place un serveur d'images supportant la décomposition hiérarchique des images.

- chargement d'images dans le système à partir de photos à haute résolution stockées dans un répertoire

- saisie et modification d'annotations
    - à partir d'un feuillet affiché
    - sélectionner la zone concernée et entrer une annotation sous forme de texte simple ou formaté
    - sélectionner une annotation existante et modifier son contenu ou la supprimer

- saisie et modification d'éléments de transcriptions

    - à partir d'un feuillet affiché
    - sélectionner la zone concernée et entrer la transcription sous forme de texte simple ou formaté
    - sélectionner une transcription existante et modifier son contenu ou la supprimer

- composition de transcriptions
    - créer une transcription à partir d'éléments de transcriptions (éventuellement sur différents feuillets)
    - modifier une transcription

De plus, le système devra être accessible sur le web sémantique des données liées (linked data) en tant que point d'accès SPARQL.

Résultats attendus

Un système permettant d'effectuer les fonctions décrites ci-dessous
Une présentation des défis techniques et la manière dont ils ont été surmontés avec les outils du web sémantique (choix technologiques par rapport à l'état de l'art, architecture du système et développement)

Évaluation

Il s'agira de démontrer, à l'aide de tests avec des utilisateurs, que le système et son interface permettent d'effectuer dans un temps raisonnable les tâches prévues (temps système + temps utilisateur). On définira pour cela un ensemble de scénarios de tâches qui seront soumis aux utilisateurs. On effectuera ces tests avec deux catégories d'utilisateurs : linguistes et grand public.

Annexe B

Script et code

B.1 Conversion d'image

B.1.1 Conversion d'un fichier TIFF en PTIF

Sur la base d'une image TIFF, on peut créer une image pyramidale (PTIF) avec l'une des commandes suivantes[1] :

Avec vips :

```
$ vips im_vips2tiff DSC_6926.TIF outputVips1.tif:deflate,tile:256x256,pyramid
```

Crée un PTIF dont toutes les pages sont compressées *lossless* avec *deflate*[2]

Avec ImageMagick :

```
$ convert DSC_6926.TIF -define tiff:tile-geometry=256x256 -compress jpeg
'ptif:outputImageMagick1.tif'
```

http://www.imagemagick.org/discourse-server/viewtopic.php?f=1&t=20193

N.B. : vips fonctionne bien et est rapide (confirmé par les auteurs de Diva.js[3]). ImageMagick ne produit pas toujours des images utilisables, et il est beaucoup moins rapide.

Pour afficher les résolutions d'une image pyramidale :

```
$ identify -verbose o.tif | grep Geometry
```

B.1.2 Conversion d'un fichier TIFF en JPEG2000

Il est aussi possible de convertir une image TIFF, sans perte, en JPEG2000[4] :

Avec kdu_compress :

```
> C:\Program Files\Kakadu\kdu_compress -i D:\DSC_6928.tif -o
D:\output.jp2 -rate -,0.5 Clayers=2 Creversible=yes Clevels=8
"Cprecincts={256,256},{256,256},{128,128}" Corder="RPCL" ORGgen_plt="yes"
ORGtparts="R" Cblk="{64,64}"
```

N.B. : kdu_compress est un programme de la librairie Kakadu. Il est gratuit, mais il n'est pas open source[5].

1. http://iipimage.sourceforge.net/documentation/images/
2. http://en.wikipedia.org/wiki/DEFLATE
3. http://journal.code4lib.org/articles/5418
4. http://iipimage.sourceforge.net/documentation/images/
5. http://iipimage.sourceforge.net/2012/05/from-scan-to-delivery-special-collections-and-iipimage-at-utrecht-university-library/

B.2 Procédures d'installation

B.2.1 Installation d'IIPMooViewer 2.0beta

Installation d'IIPMooViewer 2.0beta :

```
$ git clone https://github.com/ruven/iipmooviewer.git
```

B.2.2 Procédure d'installation d'un système Debian

- Télécharger l'un des installeurs Debian suivants :
  - AMD64 : `http://ftp.nl.debian.org/debian/dists/stable/main/installer-amd64/current/images/netboot/mini.iso`
  - i386 : `http://ftp.nl.debian.org/debian/dists/stable/main/installer-i386/current/images/netboot/mini.iso`
- N.B. : On utilise ici un "netboot" minimal avec un script preseed qui automatise l'installation. Tous les paquets nécessaires sont téléchargés durant l'installation et un script shell finalise l'installation. Durant toute l'exécution du script shell, l'installeur reste "bloqué" à 23%. À la fin, la machine s'éteint (elle ne redémarre pas).
- Graver le fichier ISO et lancer l'installation.
- Dans le menu "Installer boot menu" choisir "Advanced options".
- Dans le menu "Advanced options" choisir "Automated install".
- Choix de la langue : "French - Français". Enter.
- Choix du pays : Suisse. Enter
- Choix du clavier : Suisse Romand. Enter
- Configurer manuellement l'adresse IP, si elle n'est pas assignée automatiquement (DHCP).
  ATTENTION : Le script va formater automatiquement le (ou les) disque(s) dur(s) de la machine et toutes les données présentes seront perdues. Aucune confirmation ne sera demandée.
- Dans "Télécharger un fichier de préconfiguration", dans le champ "Location of initial preconfiguration file" :
  `http://sitehepia.hesge.ch/hepiadi/master_brero_debian_minimal_preseed_launcher.cfg`

B.2.3 Installation de IIP Server, Tomcat, OpenRDF Sésame

- A la fin de l'installation, ouvrir une session :
  - User : vmuser
  - Mot de passe : hepia
- Taper les commandes suivantes et entrer le mot de passe s'il est demandé :
  - `su`
  - `cd /root`
  - `bash auto_install_IIP_Sesame_and_website.sh`
  - Le script pose les questions suivantes :
    - "Entrez l'IP/hostname du site :"
      localhost
    - "Entrez l'adresse email du compte Admin (Usercake) (pour envoi SMTP) :"
      entre l'adresse email
    - "Entrez le username du compte email (Usercake) (pour envoi SMTP) :"
      entrer le nom du compte email
    - "Entrez le mot de passe du compte email (Usercake) (pour envoi SMTP) :"
      entrer le mot de passe du compte email
    - "Entrez le mot de passe 'root' de la base de données MySQL :"
      entrer un mot de passe quelconque
    - "Entrez l'IP ou hostname (public) du serveur pour le Remote Logger (ex : 129.194.184.101)
      entrer un adresse IP

  - "Entrez le path du Remote Logger (ex : pour hostname/logger/script.php, tapez '/logger/script.php') :"
    entrez /logging-service/remote_logger.php
  - "wait until enter key is pressed"
    Presser Enter
  - L'exécution du script est terminée.
- Convertir quelques images TIFF en JPEG2000 pour les ajouter automatiquement dans le triplestore :
  - cd /var/www/iipmooviewer
  - python auto_alimentation_4.py
- Le site est maintenant configuré et peut être consulté à l'adresse :
  http ://localhost/iipmooviewer/
  - Le nom d'utilisateur et le mot de passe administrateur sont "Admin" et "password"

B.3 Documentation pour l'installation des Triplestores

B.3.1 Allegrograph

Procédure d'installation :

- Installer la VM (on peut simplement booter sur le disque dur VMDK avec une machine crée dans VirtualBox).
  User : franz ; Pass : allegrograph
- Démarrer le serveur
- Aller sur http://localhost:10035
- "Create a repository" -> Name : "test_1"
- Cliquer ensuite sur le lien, ou aller sur : http://localhost:10035/repositories/test_1
- Charger les données "Load and Delete Data" -> "Import RDF" -> "from an uploaded file".
  - File : choisir le fichier "XYZ.txt" ; Format : Turtle ; puis "OK".

Adresse du *endpoint* : http://192.168.56.101:10035/repositories/test_1

Sources et documentation :

- http_protocol [6]
- Download et installation : [7]
- Installation depuis tar.gz [8]
  Attention : Nécessite OS 64bits
- AllegroGraph 4.8 SPARQL tutorial [9]

B.3.2 Sesame

Procédure d'installation :

- Nécessite un serveur Tomcat
- Télécharger le tar.gz ; décompresser
  - wget l'archive openrdf-sesame-2.6.9-sdk.tar.gz
  - tar -zxvf openrdf-sesame-2.6.9-sdk.tar.gz
- Copier les deux fichiers *.war dans le serveur Tomcat et démarrer Tomcat

6. http://www.franz.com/agraph/support/documentation/current/http-protocol.html
7. http://www.franz.com/downloads/clp/ag_validate_survey
8. http://www.franz.com/agraph/support/documentation/current/server-installation.html#header3-25
9. http://www.franz.com/agraph/support/documentation/current/sparql-tutorial.html

  - sudo cp openrdf-sesame-2.3.2/war/*.war /var/lib/tomcat6/webapps/
  - sudo /opt/local/share/java/tomcat6/bin/tomcatctl start
- Démarrer le serveur
- Aller sur http://localhost:8080
  - "Repository" menu -> "New Repository"
  - Type : "In Memory Store RDF Schema"
    - RDFS Inferencing in a Sesame Triplestore [10]
  - ID : "test_1"
  - Title : "Test 1"
  - Next button
  - Persist : Yes
  - Sync delay : 0
  - Create button
- Cliquer ensuite sur "Repository" menu pour voir la liste des dépôts.
- Cliquer sur le Repository fraîchement créé afin de le sélectionner.
- Dans menu "Modify" à gauche, choisir "Add"
- Un problème avec l'encodage UTF-8 semble apparaître lors du chargement d'un fichier N3. Il convient alors de le convertir en RDF/XML pour éviter ceci. La conversion peut être réalisée ici : `http://www.rdfabout.com/demo/validator/`
- Dans "RDF Data File" choisir le fichier à importer "XYZ.n3".
- Les champs Base URI et URI sont automatiquement rempli par respectivement :
  - "file://XYZ.n3"
  - "<file://XYZ.n3>"
- Bouton "upload"
- N.B. : Il peut être nécessaire de donner les droits en écriture sur le dossier de log s'il y a un problème. Il est également possible de fixer le path (basedir) et la taille de la mémoire pour Tomcat [11]

Adresse du *endpoint* : http://192.168.56.100:8080/openrdf-workbench/repositories/test_1/query

Sources et documentation :

- http_protocol [12]
- Getting Started with RDF and SPARQL Using Sesame and Python [13]
- Basic Security with HTTP authentication [14]
- Tutoriel : exemple avec l'API Sesame RDF (Java) [15]
- Dans Sesame : Getting started with Sesame (load file) [16]
- Sesame API - Adding RDF data to a repository [17]

B.3.3 Virtuoso

Procédure d'installation :

- Installer le serveur : `apt-get install virtuoso-opensource-6.1`

10. http://wes-williams.blogspot.it/2012/02/rdfs-inferencing-in-sesame-triplestore.html
11. http://timhodson.com/2011/05/installing-swiftowlim-on-macosx/
12. http://www.openrdf.org/doc/sesame2/system/ch08.html#d0e247
13. http://www.jenitennison.com/blog/node/153
14. http://rivuli-development.com/further-reading/sesame-cookbook/basic-security-with-http-authentication/
15. http://francart.fr/tutoriel-demarrage-avec-lapi-sesame-rdf/
16. http://www.snee.com/bobdc.blog/2009/02/getting-started-with-sesame.html
17. http://www.openrdf.org/doc/sesame/users/ch07.html#d0e2306

- Aller sur http://192.168.56.100:8890/conductor/
  - Login
    - user : dba (the relational data administrative account), pass : virtuoso
    - user : dav (the WebDAV adminstrative account), pass : virtuoso
- Onglet "RDF"
- Sous-onglet "RDF Store Update"
- File : choisir un fichier. ATTENTION, le N3 n'est pas accepté, utiliser une version convertie en XML. La conversion peut être réalisée ici : `http://www.rdfabout.com/demo/validator/`.

Adresse du *endpoint* : http://192.168.56.100:8890/sparql

Sources et documentation :

- Résumé d'installation, requête et load data [18]
- Wiki [19]
- Virtuoso SPARQL Query Service (HTTP) [20]
- 16.9 RDF Insert Methods in Virtuoso [21]
- 16.2 SPARQL [22]

Update de data dans Virtuoso :

"How does RDF data get into a Triple Store ?
You can import, export, and render RDF data in a number of ways :
By hand
Via HTTP/WebDAV GET and PUT
Transformation of XML to RDF via XSLT (e.g. using approaches such as GRDDL)
Transformation of SQL Data to RDF through Virtualization" [23]

"Virtuoso provides several load or data import functions that insert data into the triple store. These include :
Copy RDF files into WebDAV after mounting Virtuoso's WebDAV repository via your operating systems WebDAV filesystem mount feature Upload RDF files via the OpenLink Data Spaces Briefcase (Virtuoso automatically generates Triple Statements from WebDAV metadata) Virtuoso developers can use the Function : RDF_EXP_LOAD_RDFXML() that parses RDF/XML, Turtle, or N3 / N-Triples RDF graphs and then generates appropriate Triples Statements in Triple Store (the system table : RDF_QUAD). Direct insertion of rows into the RDF_QUAD table for nodes (RDF_QUAD_URI) or literal values (RDF_QUAD_URI_L). Mapping of Relational data from native or 3rd party heterogeneous data sources into RDF resulting in the generation of Triple Statements for Relational Entities" [24]

18. `http://kidehen.typepad.com/kingsley_idehens_typepad/2011/01/simple-virtuoso-installation-utilization-guide-for-sparql-users-update-4.html`
19. `http://www.openlinksw.com/dataspace/dav/wiki/Main/`
20. `http://virtuoso.openlinksw.com/dataspace/dav/wiki/Main/VOSSparqlProtocol`
21. `http://docs.openlinksw.com/virtuoso/rdfinsertmethods.html`
22. `http://docs.openlinksw.com/virtuoso/rdfsparql.html#rdfsparqlprotocolendpoint`
23. `http://virtuoso.openlinksw.com/dataspace/dav/wiki/Main/VOSRDFFAQ`
24. `http://virtuoso.openlinksw.com/dataspace/dav/wiki/Main/VOSRDFFAQ`

B.4 Requêtes SPARQL

Sources et documentation :

- Cambridge Semantics - SPARQL by Example (TRES BONS SLIDES) [25].
- Introduction To RDF Query With SPARQL [26].
- Exemple de commandes SPARQL Update PHP Curl INSERT, DELETE, DELETE/INSERT [27] [28] [29] [30] [31]
- Query :
  - SPARQL Guide for the PHP Developer [32]
  - SPARQL Guide for the Javascript Developer [33]
  - Count [34]
  - Count [35]
- Update (Attention : MODIFY fonctionne plus !) :
  - Syntaxe [36]
  - 4.1.3 DELETE/INSERT [37]
  - 4.1.3 DELETE/INSERT (plus récent) [38]

B.4.1 SELECT

```
SELECT ?shape ?x
FROM <file://test_001_2013-04-08.xml>
WHERE {
   ?shape rdf:type saussure:FormeGeometrique ;
            p:hasX ?x.
}
```

Listing B.1 – Retourne les formes et les valeurs de x associées

```
SELECT (count(?s) as ?c)
FROM <file://test_001_2013-04-08.xml>
WHERE {   ?s ?p saussure:ms_fr_03951_10_f032 .   }
```

Listing B.2 – Counts all SUBJECT which have a property "saussure:ms_fr_03951_10_f032"

25. http://www.cambridgesemantics.com/semantic-university/sparql-by-example
26. http://www.dajobe.org/talks/200603-sparql-stanford/
27. http://sourceforge.net/mailarchive/forum.php?thread_name=4FA82727.8030401%40ontotext.com&forum_name=sesame-general
28. http://sesame-general.435816.n3.nabble.com/sesame-SPARQL-Update-and-default-graph-td4025080.html
29. http://answers.semanticweb.com/questions/3092/insert-data-in-ontology-using-sparql
30. http://www.snee.com/bobdc.blog/2012/03/playing-with-sparql-graph-stor.html
31. http://blog.soton.ac.uk/webteam/2012/06/11/mapping-latlong-in-our-sparql-endpoint-into-uk-ordnance-survey-easting-and-northing/
32. http://www.openlinksw.com/blog/kidehen@openlinksw.com/blog/?id=1652
33. http://www.openlinksw.com/dataspace/kidehen@openlinksw.com/blog/kidehen@openlinksw.com%27s%20BLOG%20%5B127%5D/1653
34. http://stackoverflow.com/questions/5517905/sparql-query-with-count-and-order-returns-odd-result
35. http://stackoverflow.com/questions/9770807/getting-count-of-rdflist-using-sparql
36. http://sesame-general.435816.n3.nabble.com/sesame-SPARQL-Update-and-default-graph-td4025080.html
37. http://www.w3.org/TR/2010/WD-sparql11-update-20100126/#t413
38. http://www.w3.org/TR/sparql11-update/#updateLanguage

```
SELECT (count(?annot) as ?c)
FROM <file://test_001_2013-04-08.xml>
WHERE {
   saussure:ms_fr_03951_10_f032-DOT-jp2 p:contient ?zone .
   ?zone p:hasAnnotation ?annot .
}
```

Listing B.3 – Retourne le total (count) du nombre d'annotation pour ms_fr_03951_10_f032

B.4.2 INSERT

```
INSERT DATA {
   GRAPH <file://test_001_2013-04-08.xml> {
      saussure:shape_test1 rdf:type saussure:FormeGeometrique ;
                           p:hasX "42".
       }
}
```

Listing B.4 – SPARQL

B.4.3 DELETE

```
WITH <file://test_001_2013-04-08.xml>
DELETE {
   saussure:shape_test1 rdf:type saussure:FormeGeometrique ;
                        p:hasX "42".
}
WHERE {
   saussure:shape_test1 rdf:type saussure:FormeGeometrique ;
                        p:hasX "42".
}
```

Listing B.5 – Suppression des triplets

B.4.4 DELETE/INSERT/WHERE

```
WITH <file://test_001_2013-04-08.xml>
DELETE { saussure:shape_test1 p:hasX "42" . }
INSERT { saussure:shape_test1 p:hasX "43" . }
WHERE { saussure:shape_test1 rdf:type saussure:FormeGeometrique }
```

Listing B.6 – Remplace 42 par 43

B.5 Extraits complets

B.5.1 Extrait du fichier de log généré par *auto_alimentation_4.py*

```
2013-06-17 15:43:17,371 root     INFO     ################################################
2013-06-17 15:43:17,371 root     INFO     ########    Processing :
2013-06-17 15:43:17,372 root     INFO     ########    ms_fr_03951_10_f028v_029
2013-06-17 15:43:17,372 root     INFO     ################################################
2013-06-17 15:43:17,372 STDOUT   INFO     MATCHED
2013-06-17 15:43:17,372 STDOUT   INFO     ms_fr_03951_10_f028v_029
2013-06-17 15:43:17,372 STDOUT   INFO      ms_fr
2013-06-17 15:43:17,372 STDOUT   INFO      03951
2013-06-17 15:43:17,372 STDOUT   INFO      10
2013-06-17 15:43:17,372 STDOUT   INFO      f028v_029
2013-06-17 15:43:17,372 root     INFO     Testing (SPARQL ASK) :
2013-06-17 15:43:17,372 root     INFO     saussure:ms_fr_03951_10_f028v_029 rdf:type saussure:Surface_d_ecriture .
2013-06-17 15:43:17,374 root     INFO     http_code : 200  (expect code : 200)
2013-06-17 15:43:17,374 root     INFO     Return : False
2013-06-17 15:43:17,374 root     INFO     Testing (SPARQL ASK) :
2013-06-17 15:43:17,374 root     INFO     ?anything p:hasCote "ms_fr_03951_10_f028v_029" .
2013-06-17 15:43:17,375 root     INFO     http_code : 200  (expect code : 200)
2013-06-17 15:43:17,375 root     INFO     Return : False
2013-06-17 15:43:17,376 root     INFO     Testing (SPARQL ASK) :
2013-06-17 15:43:17,376 root     INFO     saussure:ms_fr_03951_10_f028v_029-DOT-jp2 rdf:type saussure:Photo .
2013-06-17 15:43:17,377 root     INFO     http_code : 200  (expect code : 200)
2013-06-17 15:43:17,377 root     INFO     Return : False
2013-06-17 15:43:17,377 root     INFO     Testing (SPARQL ASK) :
2013-06-17 15:43:17,377 root     INFO     ?anything p:hasFilename "ms_fr_03951_10_f028v_029.jp2" .
2013-06-17 15:43:17,378 root     INFO     http_code : 200  (expect code : 200)
2013-06-17 15:43:17,378 root     INFO     Return : False
2013-06-17 15:43:17,378 root     INFO     This manuscript doesn't seem to be present in the triple-store.
2013-06-17 15:43:17,379 root     INFO     Ready to add the following triples :

saussure:ms_fr rdf:type saussure:Section ;
p:hasArchiveBox saussure:ms_fr_03951 .
saussure:ms_fr_03951 rdf:type saussure:ArchiveBox ;
p:hasSubdivisions saussure:ms_fr_03951_10 .
saussure:ms_fr_03951_10 rdf:type saussure:Subdivisions ;
p:hasSurfaceEcriture saussure:ms_fr_03951_10_f028v_029 .
saussure:ms_fr_03951_10_f028v_029 rdf:type saussure:Surface_d_ecriture ;
p:hasCote "ms_fr_03951_10_f028v_029";
p:hasPhoto saussure:ms_fr_03951_10_f028v_029-DOT-jp2 .
saussure:ms_fr_03951_10_f028v_029-DOT-jp2 rdf:type saussure:Photo ;
p:hasCote "ms_fr_03951_10_f028v_029" ;
p:hasSource "http://saussure.unige.ch/bge/manuscript/ms_fr_03951_10_f028v_029.jp2" ;
p:hasFilename "ms_fr_03951_10_f028v_029.jp2" .

2013-06-17 15:43:17,379 root     INFO     Moving original file
2013-06-17 15:43:17,379 root     INFO     from
2013-06-17 15:43:17,379 root     INFO      /var/www/auto_alim/lot_test/
2013-06-17 15:43:17,379 root     INFO     to
2013-06-17 15:43:17,379 root     INFO      /var/www/auto_alim/processing/
2013-06-17 15:43:17,379 root     INFO       Converting to JPGE2000
2013-06-17 15:43:18,264 root     INFO     ################################
2013-06-17 15:43:18,264 root     INFO     # kdu_compress returncode : 0
2013-06-17 15:43:18,264 root     INFO     ################################
2013-06-17 15:43:18,264 root     INFO
2013-06-17 15:43:18,264 root     INFO     ################################
2013-06-17 15:43:18,264 root     INFO     # kdu_compress stdout :
2013-06-17 15:43:18,264 root     INFO     ################################
Copying XMP  tag info, size =    4649
Copying IPTC tag info, size =       8
Note:
    The default rate control policy for colour images employs visual (CSF)
    weighting factors.  To minimize MSE instead, specify '-no_weights'.

Generated 9 tile-part(s) for a total of 1 tile(s).
Code-stream bytes (excluding any file format) = 13,298,916 = 12.139563
bits/pel.
Layer bit-rates (possibly inexact if tiles are divided across tile-parts):
        0.499227, 12.136868
Layer thresholds:
        51293, 0
Processed using the multi-threaded environment, with
    8 parallel threads of execution (see '-num_threads')
2013-06-17 15:43:18,264 root     INFO
2013-06-17 15:43:18,264 root     INFO     KDU - OK
2013-06-17 15:43:18,265 root     INFO     Conversion successful
2013-06-17 15:43:18,265 root     INFO
2013-06-17 15:43:18,265 root     INFO     Moving original file
2013-06-17 15:43:18,265 root     INFO     from
2013-06-17 15:43:18,265 root     INFO      /var/www/auto_alim/processing/
2013-06-17 15:43:18,265 root     INFO     to
2013-06-17 15:43:18,265 root     INFO      /var/www/auto_alim/archive_originals/
2013-06-17 15:43:18,265 root     INFO     Moving JP2000 file
2013-06-17 15:43:18,265 root     INFO     from
2013-06-17 15:43:18,265 root     INFO      /var/www/auto_alim/processing/
2013-06-17 15:43:18,265 root     INFO     to
2013-06-17 15:43:18,266 root     INFO      /var/www/auto_alim/public_storage/
2013-06-17 15:43:18,266 root     INFO     Inserting in the triple-store
2013-06-17 15:43:18,595 root     INFO     http_code : 204  (expect code : 204)
2013-06-17 15:43:18,596 root     INFO     RDF Insert successful !
2013-06-17 15:43:18,596 root     INFO     !!!!!!!!!!!!!!!!!!!!!!!!!
2013-06-17 15:43:18,596 root     INFO     !        SUCCESS        !
2013-06-17 15:43:18,596 root     INFO     !!!!!!!!!!!!!!!!!!!!!!!!!
```

B.6 Formulaires HTML supplémentaires - non intégrés

Certaines des suggestions qui devaient faire partie des travaux futurs (section 8.1) ont été proposées dans le cadre d'un travail de stage à HEPIA (Haute école du paysage, d'ingénierie et d'architecture de Genève). L'auteur de ce mémoire a proposé le cahier des charges et supervisé la réalisation de deux formulaires.

Leur réalisation se basait sur l'interface uniquement. Cependant, une bonne compréhension du projet était tout de même nécessaire.

Les formulaires sont fonctionnels et permettront de faciliter certaines opérations pour les utilisateurs. Cependant, il reste encore à les intégrer dans le site. En effet, afin d'éviter d'amalgamer le travail réalisé dans ce mémoire et celui réalisé par le stagiaire, cette étape a été volontairement mise de côté pour éviter les confusions.

B.6.1 Script de correction des noms de fichiers erronés

Ce formulaire (dans le dossier http://hostname/autoalimentation) permet de renommer des fichiers qui ne correspondent pas à la convention de nommage, ou de forcer l'ajout de manuscrits dont le nom est ambigu, en découpant manuellement la cote. Il est aussi possible de supprimer un fichier si celui-ci s'est retrouvé là par erreur (par ex. : un fichier qui a déjà été publié).

L'algorithme de découpage réutilise le code existant en faisant un appel au script Python (**auto_alimentation_4.py**).

Lorsque le découpage est forcé, un fichier XML nommé comme l'image (mais avec l'extension *.xml) est généré pour spécifier quels *tokens* correspondent à quels éléments.
Par exemple **arch_saussure_396_3_012r.tif**

```
<?xml version="1.0"?>
<Cote>
  <Section>arch_saussure</Section>
  <Boite>396</Boite>
  <Enveloppe>3</Enveloppe>
  <Feuillet>f012</Feuillet>
  <Autre/>
</Cote>
```

Pour que cela fonctionne, il faudra modifier le script d'auto-alimentation. Ce dernier devra effectuer un test supplémentaire : si un fichier XML nommé comme l'image existe, alors le contenu du fichier XML est utilisé en priorité et l'étape du découpage automatique n'est pas exécutée (pour autant que le contenu du XML corresponde aux critères minimum : section, boîte, feuillet).

La sélection d'un fichier dans la liste déroulante (figure B.1) charge le formulaire (figure B.2). La partie supérieure affiche le découpage tel qu'il est détecté (figure B.3), ainsi que des explications. La partie inférieure contient les trois formulaires (figure B.4).

Le premier formulaire permet de renommer une image dont le nom est erroné (par exemple, s'il contient un caractère interdit). Il convient de respecter le découpage "Section", "Boîte", "Enveloppe" (devra être renommé en "Subdivisions") et "Feuillet". Si les éléments saisis ne correspondent pas à la convention de nommage, le formulaire ne peut pas être validé. Cependant, il est possible (mais peu recommandé) d'enlever les restrictions sur l'un des éléments.

Le deuxième formulaire permet de forcer une cote non standard (qui ne respecte pas la convention de nommage). Les exemples des figures B.3 et B.5 présentent des fichiers dont le nom est correct, mais qui ne respectent pas les conventions de nommage. Ils ont été détectés comme "ambigus" lors de l'auto-alimentation.
Ce formulaire permet donc de forcer un découpage, notamment un suffixe qui n'est pas détectable automatiquement. Il suggère au-dessus de chaque champ les *tokens* "ambigus" qui ont pu être détectés.

Le dernier formulaire (seulement le bouton) permet simplement d'effacer le fichier.

Dans l'exemple de la figure B.6, le fichier "arch_saussure_tableau.tif" devrait être corrigé dans le premier formulaire :

- **Section** : "arch_saussure"
- **Boîte** : "03965" (par exemple)
- **Subdivisions** : ""
- **Feuillet** : "tableau"

Dans l'exemple de la figure B.3, le fichier "ms_fr_03965_09_zzzz_4eme_couv_charte_regle.tif" devrait être corrigé dans le deuxième formulaire :

- **Section** : "ms_fr"
- **Boîte** : "03965"
- **Subdivisions** : "09"
- **Feuillet** : "zzzz_4eme_couv"
- **Suffixe/autre** : "charte_regle"

Dans l'exemple de la figure B.5, le fichier "ms_fr_03965_02_f047v_048a.tif" devrait être corrigé dans le deuxième formulaire :

- **Section** : "ms_fr"
- **Boîte** : "03965"
- **Subdivisions** : "02"
- **Feuillet** : "f047v_048a"
- **Suffixe/autre** : ""

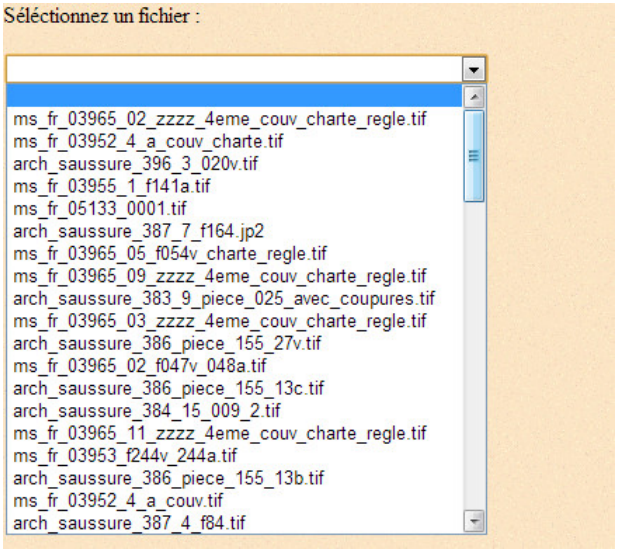


FIGURE B.1 – *Liste des fichiers problématiques*

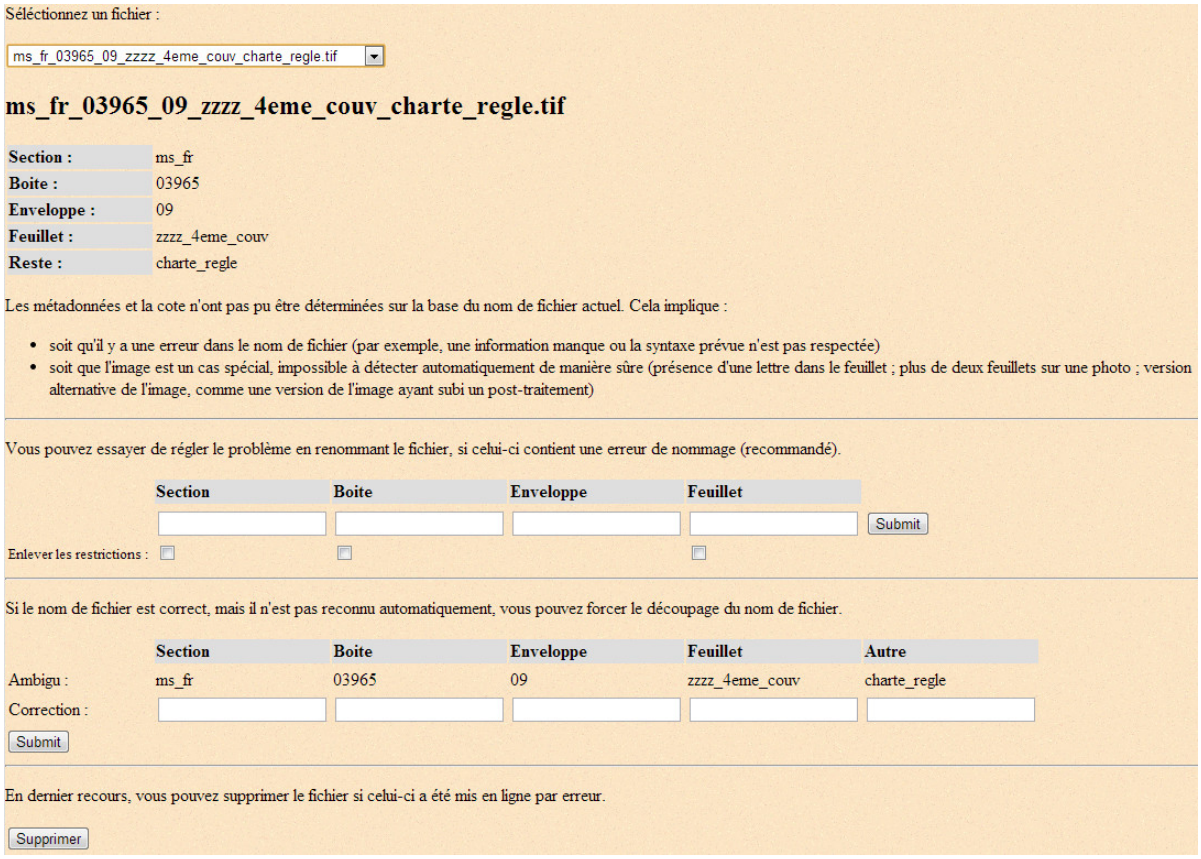


FIGURE B.2 – *Résumé du découpage automatique*

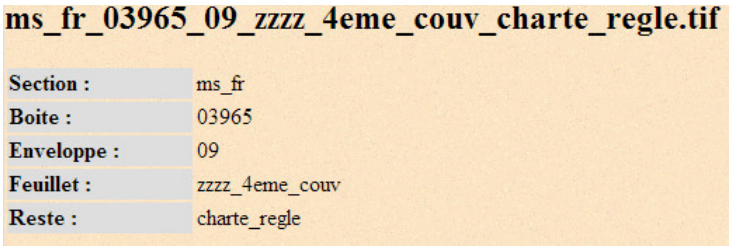


FIGURE B.3 – *Résumé du découpage d'une cote ambiguë*

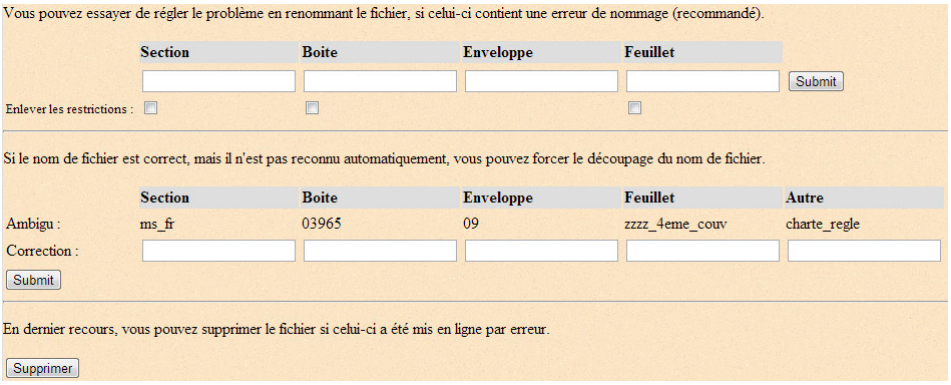


FIGURE B.4 – *Les trois formulaires*

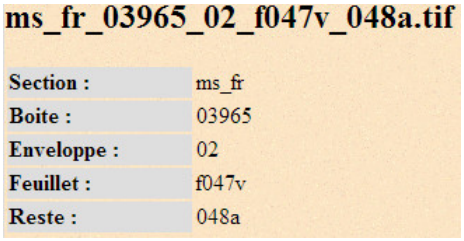


FIGURE B.5 – *Résumé du découpage d'une cote correcte, mais non acceptée*

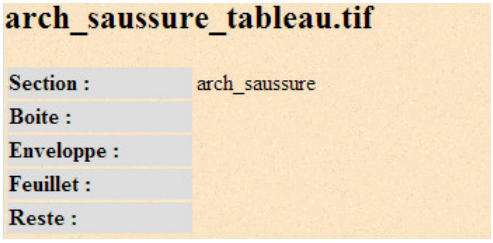


FIGURE B.6 – *Résumé du découpage d'une cote impossible*

B.6.2 Organisateur de transcriptions

Le formulaire **transcription_generator.php** est un moyen d'organiser l'ordre d'une transcription.

L'interface permet d'ajouter autant d'éléments de transcription que nécessaires à la création d'une transcription (figure B.7). Les URI complètes des éléments de transcription doivent être utilisées (figure B.8).

Il est ensuite possible de manipuler l'ordre des éléments choisis (*drag and drop*, figure B.9) et de le valider une fois terminé.

La validation n'est pas implémentée car elle nécessite des vérifications au niveau du triplestore, ainsi que la création d'une requête d'insertion (comme celle proposée dans le listing B.7).

FIGURE B.7 – *Générateur de transcription - vide*

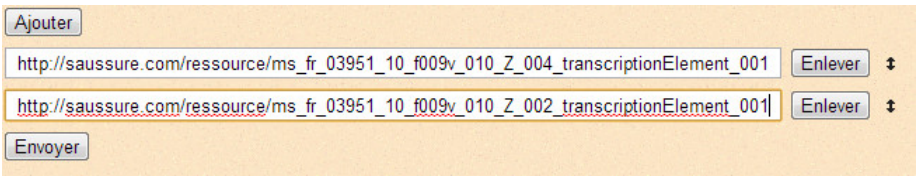


FIGURE B.8 – *Générateur de transcription - avec deux éléments*

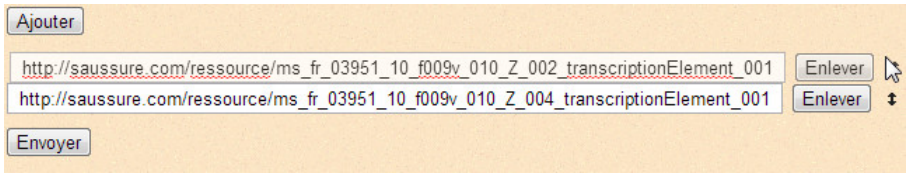


FIGURE B.9 – *Générateur de transcription - inversion des deux éléments*

```
function insertTransTemp($prefixes) {
  if( isUserLoggedIn() ) { # BEGIN AUTHENTICATION CHECK
    global $prefixes;
    global $modificationQueriesEndpoint;

    $graph_file = "<file://test_001_2013-04-08.xml>";
    // INSERT NEW ZONE + SHAPE + ANNOTATION
    $sparql_mode = "update";
    $query = $prefixes .'
    INSERT DATA {
      GRAPH '. $graph_file .' {
        saussure:testlist7 rdf:type saussure:Transcription ;
          p:malist
            [ rdf:type rdf:List;
            rdf:first saussure:ms_fr_03951_10_f031v_032_Z_001_transcriptionElement_001 ;
            rdf:rest [
              rdf:type rdf:List;
              rdf:first saussure:ms_fr_03951_10_f031v_032_Z_002_transcriptionElement_001 ;
              rdf:rest [
                rdf:type rdf:List;
                rdf:first saussure:ms_fr_03951_10_f031v_032_Z_003_transcriptionElement_001 ;
                rdf:rest [
                  rdf:type rdf:List;
                  rdf:first saussure:ms_fr_03951_10_f031v_032_Z_005_transcriptionElement_001 ;
                  rdf:rest [
                    rdf:type rdf:List;
                    rdf:first saussure:ms_fr_03951_10_f031v_032_Z_004_transcriptionElement_001 ;
                    rdf:rest rdf:nil
                  ]
                ]
              ]
            ]
            ] .
      }
    }
    ';
    $resultInsert = sparql_curl_request($sparql_mode, $modificationQueriesEndpoint, $query);
    echo $resultInsert;
  }
}
```

Listing B.7 – Requête SPARQL pour l'ajout d'une transcription

Annexe C

Notes, résumés et recherches annexes

C.1 RDF/Triplestore related

C.1.1 Liste de triplestores, d'outils et de librairies

Le W3C donne une liste des triplestores et autres librairies compatibles avec le langage de requête SPARQL [1].
Quadstore Compatibility (+liste des endpoints) [2]

Il existe quelques librairies pour l'interaction avec Sesame en PHP :
- sparllib [3]. Permet la récupération de données de plusieurs triplestore, dont Sesame.
- phpSesame [4]. La construction des requêtes n'est pas pratique, le projet n'offre aucun support et la documentation est pauvre.
- PHP/cURL - Sesame2.6 Library [5] pour Joomla. Ni code ni binaire n'est diponible !

Ces librairies n'ont pas été testées, mais sont citées par souci de complétude :
- Dead Simple : RDF and SPARQL using PHP [6] (pas de mise à jour depuis 2006)
- Redland RDF Libraries [7]
- SPARQL Javascript Library [8]
- PHP5 : Semantic Search - Web 3.0 for Drupal [9]
- RAP - RDF API for PHP [10] [N.B. : pas pour triplestore]
- Perl : RDF::AllegroGraph::Easy - Simplistic Interface to AllegroGraph HTTP server [11]
- RDF-Sesame-0.17 [12]

1. http://www.w3.org/2001/sw/wiki/index.php?title=Special:Ask&q=[[Category:Tool]][[SWTechnology::SPARQL]]&p=format=ul/template=ToolDisplayLinkWithNameWithcategoryAndLanguage/link=none/columns=1&po=?Tool+Name=%0A?Category=%0A?FProgramming+language=%0A
2. https://www.assembla.com/wiki/show/ldif/QuadStore_Compatibility
3. http://graphite.ecs.soton.ac.uk/sparqllib/
4. https://github.com/alexlatchford/phpSesame
5. http://www.emren.net/projects/library.html
6. http://blog.literarymachine.net/?p=5
7. http://librdf.org/
8. http://www.w3.org/2001/sw/wiki/SPARQL_Javascript_Library
9. http://semanticsearch.org/
10. http://www4.wiwiss.fu-berlin.de/bizer/rdfapi/
11. http://search.cpan.org/~drrho/RDF-AllegroGraph-Easy/
12. http://search.cpan.org/~mndrix/RDF-Sesame-0.17/

C.1.2 Recherche d'informations dans le triplestore

> *"[Q :] I was actually wondering how to build the index but if I think about it, maybe I need to export my RDF data in .nt or.rdf format, build the index and then maybe to synchronize the index every time RDF statements are added or edited.*
> *[A :] there are three basic ways : 1) index the RDF source. You can index RDF with solr using (if you want) the XML serialization, as it supports xpath and there are several examples online with XML that you could use as a base. 2) you could save RDF into a triplestore and then query on it and index the results, for example in solr. This way you [could] also index some inferenced result or the result of an horizontal projection. 3) you could use a triplestore which has an internal full-text index, such as lucensail/sahara, virtuoso, etc. SIREn uses sesame and a lucene index, so is half-way between 2,3"* [13]

How does indexing work on triple stores ? [14]

Sources et documentation potentiellement intéressantes :

- Squiggle : a Semantic Search Engine for indexing and retrieval of multimedia content [15]
- A New Efficient Semantic Web Platform Based on the Solr, SIREn and RDF [16]
- Indexation conceptuelle / Indexation sémantique :
  - Sur la piste de l'indexation conceptuelle de documents [17]
  - An ontology-based retrieval system using semantic indexing [18]
- SPARQL Free text search "Alternatives" [19]
- The Sesame LuceneSail : RDF Queries with Full-text Search NEPOMUK Technical Report 2008-1 [20]
- Benchmarks réalisés par le chef du projet SIREn [21]

Benchmarking Fulltext Search Performance of RDF Stores :

> *" Jena : Jena 2.5.6, ARQ 2.5.0, Lucene 2.3.1, TDB 0.6.0*
> *Sesame2 : Sesame 2.2.1, NativeStore, LuceneSail (Hits-Set) 1.3.0, Lucene 2.3.2*
> *Virtuoso : Virtuoso 5.0.9*
> *YARS : YARS post beta 3, Lucene 1.9.1*
> *All Java-based RDF stores employ the Lucene full-featured text search engine Java library, the state-of-the-art Java implementation for Information Retrieval. "* [22]

13. http://answers.semanticweb.com/questions/16143/fulltext-search-over-rdf-with-siren
14. http://answers.semanticweb.com/questions/2017/how-does-indexing-work-on-triple-stores
15. http://iricelino.org/sites/default/files/publications/2006_W06.pdf
16. http://www.ier-institute.org/2070-1918/lnit25/lnit%20v25/001.pdf
17. http://pagesperso.lina.univ-nantes.fr/~prie-y/download/prie-dni.pdf
18. http://etd.lib.metu.edu.tr/upload/12612110/index.pdf
19. http://answers.semanticweb.com/questions/8676/do-you-use-full-text-search-with-sparql-if-so-how-and-why
20. http://www.dfki.uni-kl.de/~sauermann/papers/Minack+2008.pdf
21. https://confluence.deri.ie:8443/display/SIREn/Benchmarks
22. http://livingknowledge.europarchive.org/images/publications/eswc2009minack.pdf

C.2 Modification synchrone du triplestore / Accès concurrent et atomicité en PHP

L'accès concurrent à des ressources RDF gérées par un triplestore ne permet pas systématiquement la gestion des accès concurrents à une ressource partagée, en l'occurrence un graphe RDF, bien que certains triplestores offrent une API (Java, etc.) qui gère des transactions de manière bloquante.
Le cas qui nous intéresse est celui de l'ajout de triplets dans le triplestore via un *endpoint* SPARQL.

Comme décrit dans la section 6.6, au moment de l'ajout d'une annotation le script PHP génère un identifiant unique pour la zone, la forme et l'annotation. Pour ce faire, il calcule le nombre d'annotations existantes liées à l'image (SELECT+count), l'incrémente de un et le concatène à la fin d'un nom de ressource. L'existence dans le triplestore de cette ressource est vérifiée (SPARQL ASK). Cette opération (incrément, concaténation, ASK) est répétée jusqu'à l'obtention d'une URI unique. Une fois obtenue, l'ajout peut être réalisé (SPARQL INSERT).

Cette procédure laborieuse est nécessaire, car certains mécanismes couramment présents dans les systèmes de base de données relationnelles (*triggers*, auto-incrément, génération d'UID) n'existent pas dans les triplestores. Ces opérations sont donc laissées à la charge du développeur. Cependant, même si certaines requêtes SPARQL sont atomiques, il n'est pas possible d'exécuter toutes les étapes précédentes comme une seule requête. Il y a donc un risque que des données soient inconsistantes.

Les requêtes sont générées et exécutées par un script PHP qui pourrait gérer les accès concurrents aux données (par ex. : avec des sémaphores). Malheureusement, PHP n'est pas réellement conçu pour cela et c'est généralement le système de BDD qui endosse cette responsabilité.

Prenons l'exemple de deux clients qui chargent localement les 10 annotations liées à une image (les annotations ont les identifiants ID1, ID2, ..., ID10). Deux utilisateurs distincts créent respectivement une nouvelle annotation avec le client 1 et avec le client 2 . Au moment de la validation, les deux clients envoient chacun une requête d'ajout au service frontal (section 6.6).

Les requêtes des clients sont chacune exécutées comme un processus séparé et, selon les changements de contexte lors de l'exécution sur le serveur (celui qui héberge le script PHP), il y a un risque que deux ressources avec l'ID 11 soient créées l'une à la suite de l'autre.

Grâce à l'*open world assumption* inhérent à RDF, rien n'empêche deux zones de posséder le même identifiant (il s'agira en réalité de la même zone). Comme l'identifiant de l'annotation et de la forme sont tous les deux générés sur la base de l'identifiant de la zone, chacune de ces ressources peut alors se retrouver avec des propriétés à double : une annotation avec deux propriétés p :hasText, etc. ; une forme avec deux propriétés p :hasX, deux propriétés p :hasY, etc.

Il faudrait donc que toutes les opérations ci-dessus, de la génération d'URI jusqu'à l'ajout dans le triplestore, soient effectuées comme une seule opération atomique (transaction).

Pour ce travail, il serait possible d'utiliser l'API Java de Sesame qui offre bien plus de possibilités que l'API HTTP. La solution la plus simple et la moins intrusive consiste à utiliser un sémaphore partagé en PHP :

- Un sémaphore partagé pour tous les processus (= chaque requête d'un client sur le serveur crée un nouveau processus) [23]

23. `http://www.re-cycledair.com/php-dark-arts-semaphores`

- Considération sur d'éventuels problèmes liés aux sémaphores en PHP (*What Is Wrong With PHP's Semaphore Extension*) [24]

Sources et documentation :

- ACID in Triple Stores [25]
- Pour Sesame : Sesame : transaction / Thread safe support ? [26]
- Gestion des requêtes / *multi-process* / *multi-thread* sur serveur PHP [27]
- Bonne solution pour un incrément (avec création d'une fonction SPARQL et BIND) : *How to model consecutive numbering without transactions ?* [28]
- Autres solutions possibles en PHP pour garder une information persistante au fil des requêtes :
  - PHP : Utiliser un fichier et *flock* comme un mutex. Le problème est que le verrou sur le fichier peut être ignoré par certains systèmes d'exploitation [29]. De plus il nécessite de pouvoir écrire sur le disque dur du serveur (opération lente).
  - PHP : Utiliser *Memecache* ou *Memecached* pour mettre en cache la requête vers une base de données MySQL. Exclu du fait que l'on n'utilise précisément pas de base de données [30].

24. http://jonathonhill.net/2012-12-08/what-is-wrong-with-phps-semaphore-extension/
25. http://answers.semanticweb.com/questions/11523/acid-in-triple-stores
26. http://answers.semanticweb.com/questions/11896/sesame-transaction-thread-safe-support
27. http://stackoverflow.com/questions/2921469/php-mutual-exclusion-mutex
28. http://answers.semanticweb.com/questions/17389/how-to-model-consecutive-numbering-without-transactions
29. http://stackoverflow.com/questions/3990324/why-is-locking-such-a-mess-in-php
30. http://stackoverflow.com/questions/6479328/natively-persisting-data-across-php-requests

Annexe D

Documentation technique

D.1 Accès au *endpoint* SPARQL (librairie/cURL)

D.1.1 Description des fonctions de sparql_php_curl_lib.php

Le fichier **sparql_php_curl_lib.php** propose trois fonctions qui sont des *wrappers* de la librairie curl en PHP.

- **sparql_curl_query($endpoint, $sparql_query)**
  Exécute la requête de recherche $sparql_query sur le endpoint $endpoint.
- **sparql_curl_update($endpoint, $sparql_update_query)**
  Exécute la requête de modification $sparql_update_query sur le endpoint $endpoint.
- **sparql_curl_request($mode, $endpoint, $the_sparql_request)**
  Wrapper autour des deux fonctions précédentes. Si le paramètre $mode est un *string* égal à "query", alors la fonction sparql_curl_query est appelée. Si le paramètre $mode est un *string* égal à "update", alors la fonction sparql_curl_update est appelée.

D.2 Construction et exécution des requêtes SPARQL

D.2.1 Liste des fonctions de sparql_request_pool.php

```
/**************************
**  ARCHIVES NAVIGATION
**************************/
function getTotalOfAllPhotos($prefixes, $graph_file)
function getSectionList($prefixes, $graph_file)
function getBoxFromSection($prefixes, $graph_file, $section)
function getAllInBox($prefixes, $graph_file, $box)
function getSubdivisionFromBox($prefixes, $graph_file, $box)
function getFeuilletFromSubdivisionOrBox($prefixes, $graph_file, $box, $subdivisions)
/**************************
**  SEARCH
**************************/
function searchCote($searchbool, $searchWordsArray,  $prefixes, $graph_file, $sparql_mode)
function searchConcept($conceptArray, $prefixes, $graph_file, $sparql_mode)
function prefixForRegexInSPARQL($searchWords, $mode_regex)
function searchAnnotationOr_what($searchWordsArray, $searchfor, $searchbool, $matchWhole,
    $mode_regex , $prefixes, $sparql_mode )
/**************************
**  VIEWER RELATED
**************************/
function getAnnotationOrTranscription($searchfor, $imagesfilename)
function getThumbnailsFromFilename($imagesfilename, $prefixes= "")
function getTranscriptionElementList()
```

```
/**************************
*** ADD / EDIT / DELETE
**************************/
function html2TextRegexExAndStrip($string)
function insertTransTemp($prefixes)
function addNewX($imagesfilename, $resultNewAnnotArray, $xSuffix, $xProperty, $xType)
function addNewAnnotation($imagesfilename, $resultNewAnnotArray)
function addNewConceptualAnnotation($imagesfilename, $resultNewAnnotArray)
function addNewTranscription($imagesfilename, $resultNewAnnotArray)
function editX($annotToEditArray, $xProperty, $xType)
function editAnnotation($annotToEditArray )
function editConceptualAnnotation($annotToEditArray )
function editTransElement($annotToEditArray)
function deleteX($annotToDelArray, $xProperty, $xType)
function deleteAnnotation($annotToDelArray)
function deleteConceptualAnnotation($annotToDelArray)
function deleteTransElement($annotToDelArray)
```

D.2.2 Description des fonctions de sparql_request_pool.php

La liste suivante décrit ce que retournent les fonctions de **sparql_request_pool.php** et, dans certains cas, ce qui se passe dans le code. Les fonctions sont groupées selon leur but :

- **getTotalOfAllPhotos($prefixes, $graph_file)**
  Retourne le nombre total de ressources de type saussure :Photo (nombre total d'images disponibles dans le système).
- **getSectionList($prefixes, $graph_file)**
  Retourne la liste des ressources de type saussure :Sections.
- **getBoxFromSection($prefixes, $graph_file, $section)**
  Retourne la liste des boîtes de la section $section.
- **getAllInBox($prefixes, $graph_file, $box)**
  Retourne la liste des fichiers contenus dans la boîte $box (qu'ils soient dans une subdivision ou directement dans la boîte).
- **getSubdivisionFromBox($prefixes, $graph_file, $box)**
  Retourne la liste des subdivisions ou des fichiers contenus dans la boîte $box.
- **getFeuilletFromSubdivisionOrBox($prefixes, $graph_file, $box, $subdivisions)**
  Retourne la liste des fichiers contenus dans la subdivision $subdivisions. Si le paramètre $subdivisions est égal au *string* "empty", alors la fonction retourne la liste des fichiers contenus dans la boîte $box.
- **searchCote($searchbool, $searchWordsArray, $prefixes, $graph_file, $sparql_mode)**
  Retourne la liste des fichiers et la cote à laquelle le fichier est associé.
  Le tableau $searchWordsArray est passé dans la fonction prefixForRegexInSPARQL, puis la requête de recherche est construite selon les paramètres $searchbool pour recherche "Tous" ou "Au moins un" des mots-clefs parmi les éléments de $searchWordsArray.
- **searchConcept($conceptArray, $prefixes, $graph_file, $sparql_mode)**
  Retourne la liste des fichiers associés au concept contenu dans $conceptArray (tableau d'un élément).
- **prefixForRegexInSPARQL($searchWords, $mode_regex)**
  Permet de "hacker" l'utilisation des guillemets. Si un élément de $searchWords est entouré de double guillemets ("), le contenu est gardé comme un tout. Sinon, si le terme de recherche ne contient qu'un seul double guillemet, il est préfixé pour être utilisé comme terme de recherche.
- **searchAnnotationOr_what($searchWordsArray, $searchfor, $searchbool, $matchWhole, $mode_regex , $prefixes, $sparql_mode)**
  Retourne la liste des fichiers et le texte contenant les mots-clefs.
  La requête de recherche est construite selon plusieurs paramètres. $searchfor détermine si l'on exécute la recherche dans les "annotations", les "transcriptions", ou "both" (les deux).

$searchbool détermine si l'on recherche "Tous" ou "Au moins un" des mots-clefs parmi les éléments du tableau $searchWordsArray. Si le paramètre $matchWhole est égal au *string* "WHOLE", les termes de recherche sont considérés comme des mots entiers.

- **getAnnotationOrTranscription($searchfor, $imagesfilename)**
  Retourne les annotations, les éléments de transcription ou les annotations conceptuelles. Le paramètre $searchfor peut être égal aux *strings* "annotations", "conceptualannotations", "transcriptionsElements" ou "transcriptions". Dans les trois premiers cas, la fonction retourne les coordonnées de la zone sur l'image, l'identifiant de l'objet et le texte. Dans le dernier cas ("transcriptions"), la fonction retourne les textes des transcriptions liés à l'image.
- **getThumbnailsFromFilename($imagesfilename, $prefixes= "")**
  Retourne la liste des images contenues dans la même subdivision (ou boîte s'il n'y a pas de subdivision).
- **getTranscriptionElementList()**
  Retourne la liste de tous les éléments de transcription. Utilisé dans "transcription_generator.php" pour les suggestions.
- **html2TextRegexExAndStrip($string)**
  Transforme un texte HTML en texte brut : remplace les balises
 par des \n, puis applique la fonction PHP strip_tags, et retourne le résultat en décodant les entités HTML. Cette fonction est utilisée dans addNewX, editX et deleteX.
- **insertTransTemp($prefixes)**
  Fonction de test.
- **addNewX($imagesfilename, $resultNewAnnotArray, $xSuffix, $xProperty, $xType)**
  Construit et exécute la requête d'ajout (INSERT) de type $xType. Vérifie si l'utilisateur est authentifié (voir session PHP et Usercake). Compte le nombre de zones associées à l'élément courant, puis exécute une boucle : construit une requête ASK pour vérifier si l'identifiant de la zone existe. Tant que la requête ASK retourne "vrai" (= la zone existe), alors le suffixe numérique de la zone est incrémenté (Z_001, Z_002, etc.).
  Dès qu'un identifiant unique pour la zone a été convenu, la forme et l'élément de type $xType sont créés.
  Comme des identifiants uniques sont créés, ils sont retournés au client dans du JSON afin qu'il puisse mettre à jour, par exemple, son tableau d'annotations local.
  N.B. : Cette fonction devrait être exécutée de manière atomique.
- **addNewAnnotation($imagesfilename, $resultNewAnnotArray)**
  Wrapper autour de addNewX pour l'ajout d'une annotation
- **addNewConceptualAnnotation($imagesfilename, $resultNewAnnotArray)**
  Wrapper autour de addNewX pour l'ajout d'une annotation conceptuelle
- **addNewTranscription($imagesfilename, $resultNewAnnotArray)**
  Wrapper autour de addNewX pour l'ajout d'un élément de transcription
- **editX($annotToEditArray, $xProperty, $xType)**
  Construit et exécute la requête d'édition ou d'*update* (DELETE/INSERT/WHERE) de type $xType. Vérifie si l'utilisateur est authentifié (voir session PHP et Usercake), puis remplace la zone, l'objet (annotation, élément de transcription...) et la forme par un nouveau contenu.
  N.B. : Cette fonction devrait être exécutée de manière atomique.
- **editAnnotation($annotToEditArray)**
  Wrapper autour de editX pour l'édition d'une annotation.
- **editConceptualAnnotation($annotToEditArray)**
  Wrapper autour de editX pour l'édition d'une annotation conceptuelle.
- **editTransElement($annotToEditArray)**
  Wrapper autour de editX pour l'édition d'un élément de transcription.

- **deleteX($annotToDelArray, $xProperty, $xType)**
  Construit et exécute la requête d'effacement (DELETE/WHERE) de type $xType. Vérifie si l'utilisateur est authentifié (voir session PHP et Usercake), puis supprime tous les triplets qui existent concernant cet objet.
  N.B. : Cette fonction devrait être exécutée de manière atomique.
- **deleteAnnotation($annotToDelArray)**
  Wrapper autour de deleteX pour la suppression d'une annotation.
- **deleteConceptualAnnotation($annotToDelArray)**
  Wrapper autour de deleteX pour la suppression d'une annotation conceptuelle.
- **deleteTransElement($annotToDelArray)**
  Wrapper autour de deleteX pour la suppression d'un élément de transcription.

D.3 Affichage et mise en forme des réponses des requêtes

D.3.1 Liste des fonctions de display_sparql_lib.php

```
/**************************
*** PREFIXES ALTERATION, KEYWORDS HIGHLIGHTING
**************************/
function buildPrefixesArray($prefixesToReplace)
function replacePrefixesInString($prefixesArray, $string)
function createSearchTermsArray($searchTerm)
function constructRegexWholeWord($keyword)
function constructRegex($keyword)
function highlight_paragraph($string, $keyword, $class, $findExactWord)
function my_highlightWithRegex($searchTerm, $textToModify, $findExactWord=false)
/**************************
*** DISPLAY NAVIGATION OR SEARCH RESULTS
**************************/
function printJSONAsTable($json, $searchTerm="", $query="", $matchWhole="", $customText="",
    $extraUrlParameter="", $filepath="/var/www/massimo-dev/www/auto_alim/public_storage",
    $printQueryBool=true)
function printArrayAsTable($array, $stuff="", $prefixesToReplace="", $box="", $searchTerm="",
    $query="", $matchWhole="", $customText="", $extraUrlParameter="",
    $filepath="/var/www/massimo-dev/www/auto_alim/public_storage",
    $printQueryBool=true, $highlightSearchTermsBool=true)
function printSearchResults($resultQuery, $searchTermsArray, $query, $matchWhole, $customText="")
/**************************
*** ADD / EDIT / DELETE
**************************/
function printAddNewAnnotation($resultInsert, $query, $jsonToReturn)
function printAddNewConceptualAnnotation($resultInsert, $query, $jsonToReturn)
function printAddNewTranscription($resultInsert, $query, $jsonToReturn)
function printEditAnnotation($resultEdit, $query)
function printEditConceptualAnnotation($resultEdit, $query)
function printEditTranscriptionElement($resultEdit, $query)
function printDeleteAnnotation($resultDelete, $query)
function printDeleteConceptualAnnotation($resultDelete, $query)
function printDeleteTranscriptionElement($resultDelete, $query)
/**************************
*** MISC.
**************************/
function printGetTranscriptionElementList($array)
function printGetAnnotationOrTranscription($resultQuery)
function printGetTotalOfAllPhotos($resultQuery)
```

D.3.2 Description des fonctions de display_sparql_lib.php

Le fichier **display_sparql_lib.php** contient toutes les fonctions qui permettent de mettre en forme les résultats des requêtes SPARQL retournés par les fonctions de **sparql_request_pool.php**. Ces fonctions ne sont pas automatiquement appelées, c'est à la charge du développeur d'appeler la bonne fonction.

- **buildPrefixesArray($prefixesToReplace)**
  Crée un tableau de correspondance entre préfixes et URI depuis le *string* $prefixesToReplace. $prefixesToReplace contient les prefixes SPARQL séparé par des \n.
- **replacePrefixesInString($prefixesArray, $string)**
  Remplace dans $string les préfixes du $prefixesArray par leur version courte.
- **createSearchTermsArray($searchTerm)**
  Crée le tableau des termes de recherche sur la base du *string* $searchTerm. Il sera affiché en haut des résultats de recherche.
  Si le terme est entouré de doubles guillemets, ils sont supprimés. Le terme de recherche est ensuite placé dans une balise span avec une class CSS pour la coloration.
- **constructRegexWholeWord($keyword)**
  Construit l'expression régulière (regex) lors d'une recherche "whole word" (mot entier).
- **constructRegex($keyword)**
  Construit l'expression régulière (regex) lors d'une recherche standard.
- **highlight_paragraph($string, $keyword, $class, $findExactWord)**
  Procède à la coloration du terme de recherche $keyword dans $string.
  Charge le $string dans le DOM. Crée une requête Xpath pour la recherche du mot-clef $keyword. Parcourt le DOM et ajoute un span autour du mot-clef. Le span possède la classe CSS $class.
- **my_highlightWithRegex($searchTerm, $textToModify, $findExactWord=false)**
  Applique la coloration pour chaque terme de recherche du tableau $searchTerm dans $textToModify, grâce à la fonction highlight_paragraph. Une fois le parcours terminé, les \n sont convertis en balises
.
- **printJSONAsTable($json, $searchTerm="", $query="", $matchWhole="", $customText="", $extraUrlParameter="", $filepath="/var/www/massimo-dev/www/auto_alim/public_storage", $printQueryBool=true)**
  Wrapper autour de printArrayAsTable. Décode le *string* $json (formaté en json) en tableau.
- **printArrayAsTable($array, $stuff="", $prefixesToReplace="", $box="", $searchTerm="", $query="", $matchWhole="", $customText="", $extraUrlParameter="", $filepath="/var/www/massimo-dev/www/auto_alim/public_storage", $printQueryBool=true, $highlightSearchTermsBool=true)**
  Construit un tableau HTML pour l'affichage des résultats de recherche ou la navigation. Pour les recherches, colore les termes de recherches avec my_highlightWithRegex.
- **printSearchResults($resultQuery, $searchTermsArray, $query, $matchWhole, $customText="")**
  Wrapper autour de printJSONAsTable pour l'affichage des résultats de recherche.
- **printAddNewAnnotation($resultInsert, $query, $jsonToReturn)**
  Met en forme le retour de la fonction **addNewAnnotation** (message de succès ou d'erreur retourné au client).
- **printAddNewConceptualAnnotation($resultInsert, $query, $jsonToReturn)**
  Met en forme le retour de la fonction **addNewConceptualAnnotation** (message de succès ou d'erreur retourné au client).
- **printAddNewTranscription($resultInsert, $query, $jsonToReturn)**
  Met en forme le retour de la fonction **addNewTranscription** (message de succès ou d'erreur retourné au client).
- **printEditAnnotation($resultEdit, $query)**
  Met en forme le retour de la fonction **editAnnotation** (message de succès ou d'erreur retourné au client).
- **printEditConceptualAnnotation($resultEdit, $query)**
  Met en forme le retour de la fonction **editConceptualAnnotation** (message de succès

ou d'erreur retourné au client).

- **printEditTranscriptionElement($resultEdit, $query)**
  Met en forme le retour de la fonction **editTransElement** (message de succès ou d'erreur retourné au client).
- **printDeleteAnnotation($resultDelete, $query)**
  Met en forme le retour de la fonction **deleteAnnotation** (message de succès ou d'erreur retourné au client).
- **printDeleteConceptualAnnotation($resultDelete, $query)**
  Met en forme le retour de la fonction **deleteConceptualAnnotation** (message de succès ou d'erreur retourné au client).
- **printDeleteTranscriptionElement($resultDelete, $query)**
  Met en forme le retour de la fonction **deleteTransElement** (message de succès ou d'erreur retourné au client).
- **printGetTranscriptionElementList($array)**
  Met en forme le retour de la fonction **getTranscriptionElementList()**, retourné à transcription_generator.php.
- **printGetAnnotationOrTranscription($resultQuery)**
  Transmet le retour de la fonction **getAnnotationOrTranscription** au viewer.
- **printGetTotalOfAllPhotos($resultQuery)**
  Met en forme le retour de la fonction **getTotalOfAllPhotos**, message de succès ou d'erreur retourné à homepage.php.

Annexe E

Evaluation de l'utilisabilité

E.1 Matériel d'évaluation

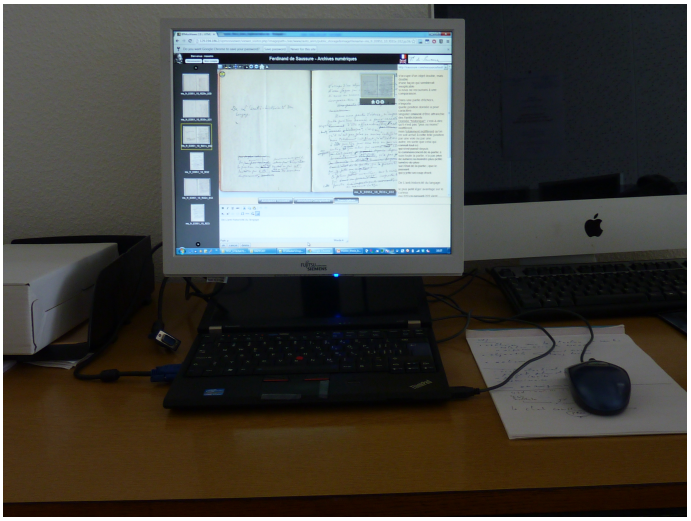


FIGURE E.1 – *Environnement de test*

Station client :
- Ordinateur portable Lenovo Thinkpad X220 (Intel Core i7-2640M @2.80GHz, 8GB RAM)
- Écran externe Dell 19" (4 :3)
- Windows 7 Professional (64 bits)
- Google Chrome Portable 26.0.1410.64 (cache, historique et cookies supprimés à chaque nouvelle session)

Serveur :
- Dell Optiplex 990 (Intel Core i7-2640M @2.80GHz, 8GB RAM)
- Debian 7 (Wheezy - Stable) (32 bits)
- Adresse publique : http ://129.194.186.2/iipmooviewer/

E.2 Protocole de test

E.2.1 Déroulement du test

Test utilisateur - déroulement

L'utilisateur est la personne effectuant le test.
L'**évaluateur** la personne qui donne les explications de départ et qui observe le déroulement du test.

- L'utilisateur remplit le questionnaire sociodémographique. (1min)
- Introduction et explications des motivations du test à l'utilisateur : quel est le but du système et son utilité, pourquoi une évaluation est nécessaire, … (2min)
- Explication du déroulement du test à l'utilisateur: (2min)
  - On teste l'interface, pas l'utilisateur. Il n'a donc pas "réussi" ou "raté" en cas de succès ou d'échec de l'un des scénarios.
  - L'utilisateur effectue quatre (4) scénarios composés d'un certain nombre de tâches. Lorsqu'une tâche est terminée, l'utilisateur doit le valider en disant « J'ai terminé ». Ceci est nécessaire pour être sûr qu'il n'a pas réalisé la tâche demandée par hasard.
  - Les textes à saisir, les concepts, et les zones sont totalement indépendantes du contenu réel.
  - Encourager l'utilisateur à réfléchir à haute voix lorsqu'il cherche quelque chose ou qu'il pense à ce qu'il doit faire. Cependant, il est important de rester concentré sur les scénarios et de ne pas dévier du sujet : les remarques sur l'interface, le fonctionnement ou le contenu seront récoltées en fin de test. Ceci afin de pouvoir valider le temps qu'a effectivement pris la tâche.
  - L'évaluateur ne doit pas intervenir ni répondre à des questions, à moins que l'utilisateur soit complétement bloqué ou pour régler des problèmes sans rapport avec l'évaluation.
- L'utilisateur lit et signe le formulaire de consentement.
- Brève démonstration de l'interface et des possibilités du système (2-3min)
- L'utilisateur peut prendre en main le système (2-3min)
- L'utilisateur effectue les scénarios dans l'ordre indiqué (voir ci-dessous). A partir de ce moment l'évaluateur n'intervient plus (sauf cas extrêmes) (10-20min).
- Lorsque tous les scénarios sont terminés, l'utilisateur remplit le questionnaire SUS (1min).
- Pour terminer, l'utilisateur est invité à donner son avis sur l'interface et à proposer d'éventuelles améliorations.

Test utilisateur - Scénarios

E.2.2 Consentement de participation

Consentement de participation

Projet	Système de visualisation et d'annotation des documents numérisés du Fond Saussure
Etudiant	Massimo Brero
Responsables	FALQUET Gilles, MARCHAND-MAILLET Stéphane, NERIMA Luka

- J'ai reçu des informations sur ce projet de recherche
- Je comprends ce projet de recherche et mon rôle dans son évaluation.
- J'ai le droit de me retirer à tout moment de cette évaluation.
- Mes données personnelles collectées pendant cette évaluation restent confidentielles.
  Dans le cas de leur publication, elles seront présentées de façon anonyme.

Je confirme avoir lu et compris les affirmations ci-dessus,

Nom du participant : ……..

Signature : ……..

Date : ……..

E.2.3 Formulaire sociodémographique et formulaire de compétences

Questionnaire sociodémographique

Nom _____ Date _____

Prénom _____ Profession _____

Age _____ Sexe _____

Niveau académique (+ titre et date d'obtention) : _____

Système d'exploitation et navigateur habituel : _____

Formulaire de compétences

	Pas du tout d'accord				Tout à fait d'accord
1. Je suis à l'aise avec un ordinateur.	☐ 1	☐ 2	☐ 3	☐ 4	☐ 5
2. J'utilise au moins une fois par semaine un ordinateur dans le cadre de mon travail ou dans ma vie privée.	☐ 1	☐ 2	☐ 3	☐ 4	☐ 5
3. Je sais utiliser le navigateur web *Google Chrome* ou *Mozilla Firefox*	☐ 1	☐ 2	☐ 3	☐ 4	☐ 5
4. J'ai des bonnes connaissances en linguistique.	☐ 1	☐ 2	☐ 3	☐ 4	☐ 5
5. J'ai étudié ou je connais la linguistique selon Ferdinand de Saussure.	☐ 1	☐ 2	☐ 3	☐ 4	☐ 5
6. Je suis un spécialiste de Ferdinand de Saussure.	☐ 1	☐ 2	☐ 3	☐ 4	☐ 5
7. J'ai déjà consulté ou étudié des manuscrits	☐ 1	☐ 2	☐ 3	☐ 4	☐ 5
8. J'étudie ou je travaille régulièrement avec des manuscrits.	☐ 1	☐ 2	☐ 3	☐ 4	☐ 5
9. Je suis au courant de ce projet et je l'ai déjà utilisé ou vu en action.	☐ 1	☐ 2	☐ 3	☐ 4	☐ 5
10. (Pour les hommes) J'aimerais avoir une moustache comme Ferdinand de Saussure.	☐ 1	☐ 2	☐ 3	☐ 4	☐ 5

E.2.4 Formulaire SUS

System Usability Scale (SUS)

	Pas du tout d'accord				Tout à fait d'accord
1. Je pense que je vais utiliser ce service fréquemment	1	2	3	4	5
2. Je trouve ce service inutilement complexe	1	2	3	4	5
3. Je pense que ce service est facile à utiliser	1	2	3	4	5
4. Je pense que j'aurai besoin de l'aide d'un technicien pour être capable d'utiliser ce service	1	2	3	4	5
5. J'ai trouvé que les différentes fonctions de ce service ont été bien intégrées	1	2	3	4	5
6. Je pense qu'il y a trop d'incohérence dans ce service.	1	2	3	4	5
7. J'imagine que la plupart des gens seraient capable d'apprendre à utiliser ce service très rapidement.	1	2	3	4	5
8 J'ai trouvé ce service très lourd à utiliser.	1	2	3	4	5
9. Je me sentais très en confiance en utilisant ce service.	1	2	3	4	5
10. J'ai besoin d'apprendre beaucoup de choses avant de pouvoir utiliser ce service.	1	2	3	4	5

Remarques / Commentaires / Propositions

……

……

……

……

……

……

E.2.5 Scénario 1

SCENARIO 1

Trouver un manuscrit donné par sa cote (recherche textuelle).

- Aller sur la page d'accueil.
- Utiliser la "**Recherche par cote**" pour chercher le manuscrit ayant la cote : **arch_saussure_368_4_f43**
- Visualiser le manuscrit **arch_saussure_368_4_f43**

Créer une nouvelle annotation textuelle.

- Vous êtes sur le manuscrit **arch_saussure_368_4_f43**
- Créer une **annotation textuelle** en dessinant une zone rectangulaire autour du mot « **Monsieur** » en haut de la page. N'hésitez pas à zoomer sur l'image.
- Dans l'éditeur taper le texte : « **Ceci est le mot Monsieur**», puis sauver l'annotation.

E.2.6 Scénario 2

SCENARIO 2

Trouver un manuscrit donné grâce à son emplacement physique à la bibliothèque (section, boîte, …).

- Aller sur la page d'accueil.
- Utiliser l'exploration/navigation des "**Archives numériques**" pour trouver le manuscrit :
  1. Afficher la liste des documents de la section **arch_saussure**, boîte **368**, subdivision **4**
  2. Choisir de visualiser le document : **f44v** (arch_saussure_368_4_f44v)
  3. Revenir à la page d'accueil
  4. Afficher la liste de tous les feuillets de la boîte **369** de la section **arch_saussure**
  5. Choisir de visualiser le document **f003** de la subdivision **8**
- Visualiser le manuscrit **arch_saussure_369_08_f003**

Lier un concept au manuscrit.

- Vous êtes sur le manuscrit **arch_saussure_369_08_f003**
- Lier le concept "**acte linguistique**" au manuscrit en dessinant une zone rectangulaire autour du bloc de texte qui se trouve en haut de la page de gauche.
- Choisir le concept "**acte_linguistique**" dans la liste, puis sauver le concept.

Changer de manuscrit lors de la visualisation d'un manuscrit en utilisant la navigation par miniatures (thumbnails).

- Vous êtes sur le manuscrit **arch_saussure_369_08_f003**.
- En utilisant la navigation par miniatures (*thumbnails*) sur la gauche, changer le manuscrit courant par **arch_saussure_369_08_f005v**.
- Visualiser le manuscrit **arch_saussure_369_08_f005v**.

En visualisant un manuscrit donné, supprimer une annotation existante.

- Vous êtes sur le manuscrit **arch_saussure_369_08_f005v**
- Double-cliquer sur l'annotation contenant le texte :
  « **Ceci n'est pas un texte de Saussure.**
  Merci d'effacer cette annotation.».
- Supprimer l'annotation

Créer une nouvelle annotation textuelle (texte mis en forme).

- Vous êtes sur le manuscrit **arch_saussure_369_08_f005v**
- Zoomer deux fois sur l'image, puis se déplacer pour afficher le bas de la page de gauche.
- Créer une **annotation textuelle** en dessinant une zone rectangulaire autour du mot « **historique** » en bas de la page de gauche.
- Dans l'éditeur taper le texte : « **historique** » (souligné), puis sauver l'annotation.

E.2.7 Scénario 3

SCENARIO 3

Trouver la liste des manuscrits liés à un concept sémantique.

- Aller sur la page d'accueil.
- Utiliser la "**Recherche par concept**" pour trouver un manuscrit lié au concept "**signifié**"
- Visualiser successivement tous les manuscrits trouvés.
  N.B. : Si le message *Confirm Form Resubmission* s'affiche, il suffit de recharger la page du navigateur (ou presser la touche F5)

Créer une nouvelle transcription.

- Vous êtes sur le manuscrit **ms_fr_03951_10_f033v_034**
- Créer une **transcription** en dessinant une zone rectangulaire autour du premier paragraphe sur la page de gauche (dont le texte est écrit en bleu-violet).
- Taper le texte suivant (ne pas déchiffrer, taper le texte fourni, tel quel) :
  « **Il n'y a de langue** » , puis sauver la transcription.

Créer une nouvelle transcription.

- Vous êtes sur le manuscrit **ms_fr_03951_10_f033v_034**
- Zoomer deux fois sur le manuscrit.
- Créer une **transcription** en dessinant une zone rectangulaire autour des trois (3) premières lignes de la page de droite. Il est nécessaire de **cacher la fenêtre de navigation** pour y arriver.
- Taper le texte suivant (ne pas déchiffrer, taper le texte fourni, tel quel) :
  « **De même tel produit** » , puis sauver la transcription.

E.2.8 Scénario 4

SCENARIO 4

Trouver un manuscrit via une recherche textuelle (mots-clefs) sur les transcriptions et les annotations existantes.

- Aller sur la page d'accueil.
- Utiliser la "**Recherche par contenu**" pour chercher le(s) manuscrit(s) contenant le mot-clef : « **échec** »
- Constater qu'il y a plusieurs résultats.
- Corriger la recherche pour trouver le(s) manuscrit(s) contenant les mots-clefs : « **échec** » et « **partie** »
- Dans les résultats, choisir le manuscrit **ms_fr_03951_10_f031v_032**
- Visualiser le manuscrit **ms_fr_03951_10_f031v_032**

Modifier une transcription existante.

- Vous êtes sur le manuscrit **ms_fr_03951_10_f031v_032**
- Prendre connaissance de la transcription sur la droite. Relever le mot **inexplicable**.
- Double-cliquer sur la zone de l'image dont la transcription est : « **s'occupe d'un objet double, mais double**
  d'une façon qui semblerait inexplicable
  si nous ne recourions à une comparaison. ».
- Remplacer le mot « **inexplicable** » par « **inextricable** » puis sauver la modification.
- Recharger la page du navigateur (ou presser la touche F5)
- Constater dans la transcription qui apparaît sur la droite que le texte a été corrigé.